Universidade de Aveiro Departamento de Matemática
2008

Diana Xavier de Almeida

O problema do caminho mais curto com restrições de capacidade

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Matemática e Aplicações, realizada sob a orientação científica do Dr. Agostinho Miguel Mendes Agra, Professor Auxiliar do Departamento de Matemática da Universidade de Aveiro

Ao meu marido, pelas eternas discussões acerca destes assuntos. Também pela companhia, apoio e compreensão.
Aos meus pais e ao meu irmão.

o júri

Presidente

Prof. Dr. Domingos Moreira Cardoso
professor catedrático da Universidade de Aveiro

Prof. Dr. Miguel Fragoso Constantino
professor auxiliar da Faculdade de Ciências da Universidade de Lisboa

Prof. Dr. Agostinho Miguel Mendes Agra
professor auxiliar da Universidade de Aveiro

agradecimentos Ao meu orientador, Dr. Agostinho Miguel Mendes Agra, pela disponibilidade, imensa dedicação e grande apoio.

palavras-chave Caminho mais curto, capacidades, relaxação.

Resumo

Neste trabalho estuda-se o problema do caminho mais curto com capacidades (PCMCRC). O PCMCRC é uma variante do problema do caminho mais curto onde existe uma restrição de capacidade associada aos arcos. Este problema tem variadas aplicações, nomeadamente na área das telecomunicações e no planeamento de rotas de veículos. Na sua forma geral o PCMCRC é $\mathcal{NP}$-difícil. É feita uma descrição do problema, uma breve referência às principais técnicas de resolução e é proposto um novo algoritmo heurístico baseado na relaxação da restrição de capacidade. É efectuado um estudo computacional com o objectivo de identificar as instâncias mais difíceis do PCMCRC e, também, de testar o novo algoritmo.

Keywords Shortest path, capacities, relaxations

Abstract

This work studies the shortest path problem with capacities (SPPC). The SPPC is a variation of the shortest path problem, where there is a capacity constraint associated with the arcs. This problem has multiple applications in areas such as telecommunications and traffic routing planning. In it's general form, it's a $\mathcal{NP}$-hard problem.

It is made a description of the problem, a slight reference to the main resolution techniques, and it's proposed a new heuristic algorithm, based on the relaxation of the capacity constraint. It is reported a computational study in order to identify the hard instances for the SPPC and in order to test the new algorithm.

Índice

1 Introdução

O problema do caminho mais curto (PCMC) ocupa um lugar de destaque no âmbito da investigação operacional. Para se entender o porquê, basta pensar-se nas vezes em que é necessário enviar alguma entidade – pessoas, objectos, veículos, mensagens, etc. – de um ponto para o outro numa determinada rede da maneira mais eficiente. Normalmente a eficiência prende-se com questões monetárias (custos), distâncias, tempo, mas muitas outras variáveis podem ser tidas em conta.

Nesta tese consideramos o PCMC com uma restrição de capacidade nos arcos, onde se assume que a utilização de cada arco implica o consumo de um recurso cuja capacidade máxima não pode ser excedida. Este recurso pode ser combustível, quando se pretende limitar o seu consumo, pode ser tempo, quando existe prazo para entrega de algo, pode servir, simplesmente, para garantir a qualidade de um determinado serviço, etc.

Ao contrário do que acontece com o PCMC, para o qual existem algoritmos eficientes para a sua resolução (Ahuja et al. 1993) quando não existem circuitos de peso negativo, o problema do caminho mais curto com restrições de capacidade (PCMCRC) é NP-difícil, mesmo considerando custos não negativos nos arcos.

Nos últimos anos, muitos têm sido os trabalhos desenvolvidos, de diferentes áreas (Computação, Investigação Operacional, etc.), com vista à resolução de problemas do caminho mais curto com restrição de capacidade. Uma técnica usual e que irá ser explorada neste trabalho, consiste em proceder à relaxação da restrição de capacidade e incluir, de algum modo, a informação dessa restrição na função objectivo. Um exemplo clássico dessa técnica é a relaxação lagrangeana.

Nesta tese é efectuada uma abordagem do problema do caminho mais curto com uma restrição de capacidade. São descritos alguns métodos propostos para a sua resolução e é introduzido um novo algoritmo baseado na relaxação da restrição de capacidade do problema que, partindo de um caminho inicial, determina sucessivamente vários caminhos até obter uma solução admissível para o problema, isto é, que respeite a restrição de capacidade, ou até atingir um número máximo (estabelecido) de iterações. Este novo

algoritmo baseia-se na ideia de relaxar a restrição de recurso, incluindo, de algum modo, na função objectivo a informação da restrição relaxada, penalizando apenas os arcos no caminho caso este não seja admissível.

É apresentado um estudo computacional que tem dois objectivos:

(i) Seguindo o estudo de Pisinger (2005) para o problema do Saco-Mochila tentar-se-á compreender quais as instâncias mais difíceis do PCMCRC.

(ii) Estudar computacionalmente o desempenho do algoritmo introduzido.

No capítulo 2 é feita uma breve referência à teoria da complexidade e a conceitos elementares da teoria dos grafos.

No capítulo 3 descreve-se o problema do caminho mais curto sem restrições e analisam-se algoritmos propostos para a sua resolução.

No capítulo 4 introduz-se e formula-se o problema do caminho mais curto com uma restrição de capacidade.

No capítulo 5 descrevem-se alguns dos mais usuais métodos e algoritmos propostos para a resolução do PCMCRC, nomeadamente um algoritmo de programação dinâmica e alguns métodos baseados em diferentes relaxações.

Um novo algoritmo heurístico é introduzido no capítulo 6. Este algoritmo determina sucessivamente vários caminhos até obter uma solução admissível para o problema e baseia-se na relaxação da restrição de capacidade.

É apresentado, no capítulo 7, um estudo computacional que se destina a identificar as instâncias mais difíceis do PCMCRC e, adicionalmente, testar um novo algoritmo.

Finalmente, no capítulo 8, são apresentadas algumas conclusões.

2 Conceitos básicos

Neste capítulo serão abordados alguns conceitos fundamentais para o desenvolvimento de todo o trabalho. Começar-se-á por analisar a questão da complexidade de problemas, passando-se, de seguida, à definição de alguns conceitos básicos de teoria dos grafos.

2.1 Complexidade

Neste capítulo far-se-á referência à teoria da complexidade, com vista à classificação de problemas.

Um algoritmo pode ser entendido como um conjunto ordenado de procedimentos que levam à resolução de um problema. Podem existir vários algoritmos para a resolução de um problema, sendo possível a sua comparação. Essa comparação poderá ser efectuada em termos de eficiência

A eficiência de um algoritmo está relacionada com o tempo gasto na execução do algoritmo e com a quantidade de memória que é necessária para executá-lo. Assim, um algoritmo será mais eficiente do que outro na resolução de um problema se obtém uma solução para o problema em menos tempo e utiliza menos espaço para representar os dados do problema (instância). Visto ser bastante difícil e, por vezes, até impossível prever com rigor o tempo de execução de um algoritmo, uma solução possível passa por majorar o número de operações elementares que o algoritmo executará para uma qualquer instância do problema em causa. Uma notação bastante utilização é a notação O, que se define da seguinte maneira:

Diz-se que f é $O(g(n))$ se $\exists c \in IR^{+}, \exists n_0 \in IN : |f(n)| \leq c|g(n)|, \forall n \geq n_0$.

Esta notação é utilizada quando se pretende um limite superior, pois ao ser utilizada permite descrever o tempo de execução de um algoritmo analisando, apenas, a estrutura do algoritmo para o pior caso. Definem-se, então, classes de complexidade algorítmica, como

por exemplo, por ordem crescente de esforço: $O(1), O(\log n), O(n), O(n^2), ..., O(n!), O(n^n)$, sendo mais eficiente, por exemplo, um algoritmo de classe de complexidade algorítmica $O(\log n)$ do que um de classe $O(n^2)$. Os algoritmos de complexidade $O(a^n), a > 1$ são algoritmos de complexidade exponencial. Na prática, terão interesse, essencialmente, algoritmos de complexidade polinomial (também denominados por algoritmos eficientes).

Entre os problemas que podem ser resolvidos por algoritmos encontram-se os problemas de decisão, para os quais a teoria da complexidade foi inicialmente desenvolvida. A cada problema de optimização está associado um problema de decisão que consiste em formular uma questão em termos de valor da função objectivo do problema de optimização.

Um problema de decisão é classificado como fácil se é possível desenvolver um algoritmo que resolva cada instância do problema em tempo polinomial.

A classe dos problemas para os quais existe um algoritmo polinomial que resolve qualquer instância do problema designa-se por $\mathcal{P}$.

Existe uma classe "mais geral" de problemas que inclui os problemas em $\mathcal{P}$. Trata-se da classe $\mathcal{NP}$ que se define como a classe dos problemas de decisão tais que para toda a instância que tenha resposta SIM, existe uma prova em tempo polinomial do SIM. A designação $\mathcal{NP}$ tem origem em "solvable by a **N**ondeterministic Turing machine in **P**olinomial time"; consultar Garey and Johnson (1979).

O conjunto dos problemas aos quais podem ser reduzidos, em tempo polinomial, cada um dos problemas da classe $\mathcal{NP}$ designa-se por $\mathcal{NP}$-Completo. Se um problema desta classe for resolúvel em tempo polinomial, então também os outros o serão. Para os problemas desta classe não é conhecido nenhum algoritmo eficiente. Conjectura-se, porém, que este tipo de algoritmos não existe, havendo, então, a distinção entre a classe de problemas $\mathcal{P}$ e $\mathcal{NP}$ (ver Figura 1). A classe dos problemas $\mathcal{NP}$-Difícil é constituída pelos problemas de optimização cujo correspondente problema de decisão está na classe $\mathcal{NP}$-Completo (Wolsey 1998).

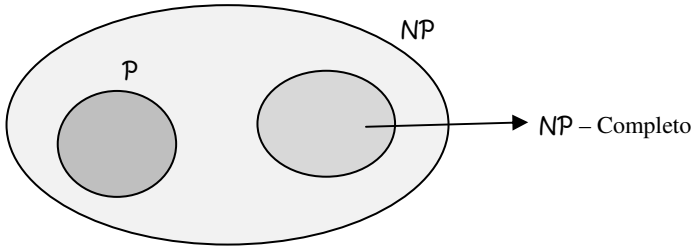


Figura 1 – Classes de complexidade

2.2 Grafos e Redes

São apresentados, de seguida, alguns conceitos básicos da teoria de grafos fundamentais para o desenvolvimento da tese, bem como a notação a utilizar neste âmbito.

Um grafo orientado (digrafo) G é um par ordenado (V, A), sendo $V = \{0,...,n\}$ um conjunto de vértices (ou nós) e A um conjunto de arcos. Na figura 2 pode observar-se um exemplo.

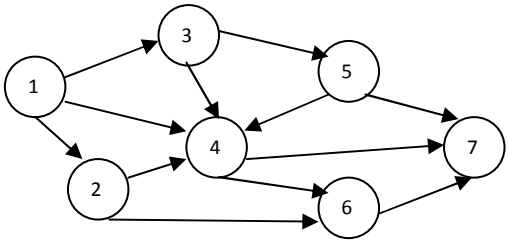


Figura 2 – exemplo de um grafo

Cada arco liga dois vértices, podendo definir-se como um par ordenado de vértices (i, j), dizendo-se, nesse caso, que o arco é divergente de i e convergente em j, sendo i antecessor de j e j sucessor de i. Designa-se por $V^+(i)$ o conjunto dos sucessores de i e por $V^-(i)$ o conjunto dos antecessores de i.

Um grafo não orientado é definido do mesmo modo que o grafo orientado, com a excepção dos arcos, que aqui são considerados conjuntos não ordenados de pares de vértices. Neste caso é usual designá-los por arestas. Um exemplo de um grafo não orientado é o que se segue, na Figura 3.

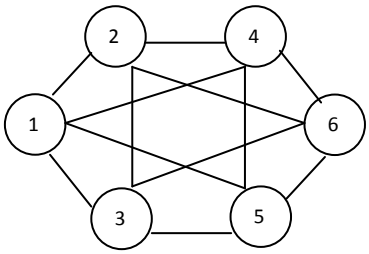


Figura 3 – exemplo de grafo não orientado

Nesta dissertação irão ser considerados apenas grafos orientados.

Um grafo (V', A') é um subgrafo de (V, A) se $V' \subseteq V$ e $A' \subseteq A$.

Um passeio num grafo $G = (V, A)$ é um subgrafo de G que é constituído por uma sequência de vértices e arcos $i_1 - a_1 - i_2 - a_2 - ... - i_{r-1} - a_{r-1} - i_r$, com $a_k \in A$. No caso de um digrafo, e por conveniência de notação, assume-se que $a_k = (i_k, i_{k+1})$. Por exemplo, no caso do grafo da figura 2, 1-4-5-7 não é um passeio, mas 1-4-6-7 é um passeio tal como se pode observar na figura 4.

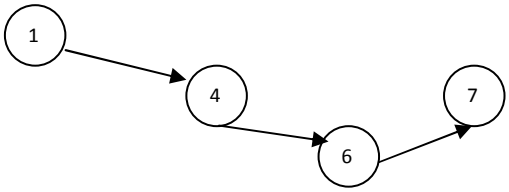


Figura 4 – exemplo de um passeio

Um trajecto é um passeio que não repete arcos.

Um caminho é um passeio sem repetição de nós, exceptuando-se, eventualmente, os nós inicial e final. Assim, num grafo, existe um caminho entre um nó origem, s, e um nó destino, t, se existe uma sequência de nós e arcos (ou arestas) que une os dois nós sem que haja repetição de nós.

Um circuito, ou trajecto fechado, é um trajecto com pelo menos uma aresta e tal que o nó inicial coincide com o nó final.

Define-se ciclo como um caminho fechado.

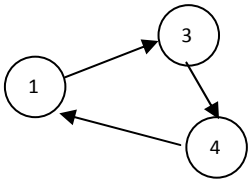


Figura 5 – exemplo de ciclo

Um grafo (digrafo) diz-se acíclico se não contiver ciclos.

Quando existem arcos paralelos entre dois vértices ou quando existem lacetes (arcos de um vértice para ele próprio) o grafo designa-se por multigrafo.

Um grafo pode ter associado aos seus arcos (ou às suas arestas) valores numéricos que indiquem, por exemplo, distâncias ou custos. Neste caso diz-se que se trata de uma rede. Uma rede é, então, um terno (V, A, C) sendo V um conjunto de vértices, A um conjunto de arcos e $C = [c_{ij}]$ a matriz das distâncias de cada ligação $(i, j) \in A$.

3 Problema do Caminho mais curto

Neste capítulo é apresentado o problema do caminho mais curto e dois algoritmos para a sua resolução. O algoritmo de Dijkstra para o caso dos custos serem não negativos, e o algoritmo de correcção de etiquetas para o caso de custos arbitrários em grafos sem ciclos de peso negativo.

Considere-se uma rede (V, A, C) onde $V = \{0,...,n\}$ é o conjunto de vértices, A o conjunto de arcos e $C = [c_{ij}]$, onde c_{ij} é a distância da ligação $(i, j) \in A$. O problema do caminho mais curto (PCMC) consiste em, dada uma rede e escolhido um dos seus vértices como origem, encontrar o caminho mais curto do vértice origem para cada um dos restantes vértices (ou, alternativamente, até um vértice destino).

3.1 Formulação do Problema

O PCMC pode ser formulado em termos de programação linear inteira.

Definindo variáveis de decisão, x_{ij}, para as ligações $(i, j) \in A$ de tal modo a que o seu valor seja 1 caso o arco (i, j) faça parte do caminho mais curto ou 0 caso contrário, isto é:

$$x_{ij} = \begin{cases} 1 & se \quad o \quad arco \quad (i,j) \quad está \quad na \quad solução \\ 0 & caso \quad contrário \end{cases}$$

O problema do caminho mais curto (PCMC) entre o vértice origem, *s*, e o vértice destino, *t*, pode ser formulado da seguinte maneira:

$$(PCMC) \qquad Min \quad \sum_{(i,j)\in A} c_{ij} x_{ij}$$

$$s.a:$$

$$\sum_{j\in V^+(s)} x_{sj} = 1 \qquad (1)$$

$$\sum_{i\in V^-(j)} x_{ij} = \sum_{i\in V^+(j)} x_{ji},\ j \in V \setminus \{s,t\} \qquad (2)$$

$$\sum_{j\in V^-(t)} x_{jt} = 1 \qquad (3)$$

$$x_{ij} \in \{0,1\}, \forall (i,j) \in A \qquad (4)$$

A igualdade (1) garante que do nó origem sai apenas um arco. Para cada nó *j* verifica-se conservação de fluxo (2). A igualdade (3), por sua vez, garante que chega um só arco ao nó destino. A desigualdade (4) garante que as variáveis de decisão x_{ij} tomam apenas o valor 0 ou o valor 1, nas condições anteriormente referidas.

Note-se que esta formulação não impede a ocorrência de soluções que incluam ciclos e que, portanto, não são caminhos, conforme se pode observar na figura 6.

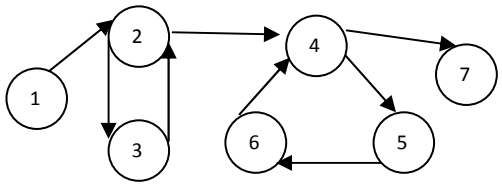


Figura 6 – exemplo de um passeio com ciclos

A garantia da inexistência de soluções que incluam ciclos é dada pelo conjunto de restrições:

$$\sum_{i\in S}\sum_{j\in S} x_{ij} \leq |S| - 1, \quad S \subseteq V, S \neq \phi \qquad (5)$$

Repare-se que, sendo $|S| = n$, um circuito envolvendo todos os vértices em S contém, pelo menos, n arcos. O conjunto de restrições (5) garante a existência de, no máximo, $n-1$ arcos para o subconjunto S de nós. Considerando todo o subconjunto de nós S, com $S \neq V$ e $S \neq \phi$ garante-se que não ocorrem ciclos na solução.

Como se pode observar nos exemplos da Figura 7, existem tantos arcos como nós, permitindo a existência de ciclos:

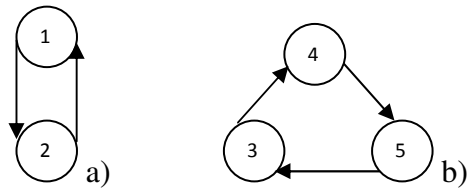


Figura 7 – ciclos

Assumindo que não existem arcos de peso negativo ou que o grafo é acíclico, o conjunto de restrições (5) pode ser ignorado.

3.2 Algoritmo de Dijkstra

Um dos algoritmos que tem vindo a ser estudado em profundidade e que resolve eficazmente o PCMC quando os pesos dos arcos são não negativos é o algoritmo de Dijkstra.

O algoritmo de Dijkstra encontra o caminho mais curto de um nó origem para qualquer outro nó de uma rede cujas distâncias são não negativas. Baseia-se na distinção entre dois conjuntos de nós: os que possuem uma etiqueta permanente que lhes garante a distância do caminho mais curto da origem a esse vértice, e os que possuem uma etiqueta temporária cujo valor é igual à distância do caminho mais curto da origem a esse vértice usando apenas como nós intermédios os nós de etiqueta permanente. A etiqueta temporária é, assim, um majorante para a distância do caminho mais curto da origem a esse vértice.

Inicialmente o algoritmo atribui uma etiqueta definitiva ao nó origem com o valor 0, e atribui uma etiqueta temporária aos restantes nós com o valor $+\infty$.

O algoritmo tornará etiquetas temporárias em etiquetas definitivas até que todos os nós tenham etiqueta definitiva ou, alternativamente, até que o nó destino tenha etiqueta definitiva.

Considerem-se:

- (V, A, C): uma rede, sendo V um conjunto de vértices, A um conjunto de arcos e $C = [c_{ij}]$ a matriz das distâncias das ligações (i, j).
- s : nó origem.
- S : conjunto dos nós com etiqueta definitiva.
- $\overline{S}$: conjunto dos nós com etiqueta temporária.
- $d(i)$: distância do caminho mais curto do nó origem, s, para o nó i, usando, apenas, nós intermédios em *S*.
- $ant(i)$: antecessor de *i* no caminho mais curto do vértice *s* para o vértice *i.*

Algoritmo CMC

Input: (V, A, C), s, $n = |V|$

Output: d, ant

Inicialização:

$d(i) = +\infty, \quad i \in V \setminus \{s\}$

$d(s) = 0$

$ant(s) = s$

$S = \{s\}$

$\overline{S} = V \setminus \{s\}$

(i) Enquanto $|S| < n$, *fazer:*

Seja $i \in \overline{S}$ *o nó para o qual* $d(i) = \min\{d(j), j \in \overline{S}\}$

$S = S \cup \{i\}$

$\overline{S} = \overline{S} \setminus \{i\}$

Para cada $j \in V^+(i) \setminus S$ *fazer:*

Se $d(j) > d(i) + c_{ij}$, *então:*

$d(j) = d(i) + c_{ij}$

$ant(j) = i$

Este algoritmo permite resolver problemas do caminho mais curto entre um nó origem s e todos os restantes nós. No caso de se ter um nó t estipulado como nó destino, deverá substituir-se *(i)* no algoritmo anterior por *(ii)*:

(ii) *Enquanto* $t \in \overline{S}$, *fazer:*

3.2.1 Exemplo

Considere-se a rede da figura 7, na qual se pretende determinar o caminho mais curto entre o nó origem *1* e o nó destino *5*.

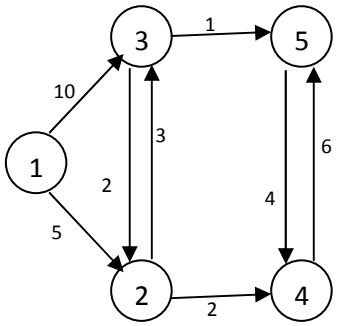
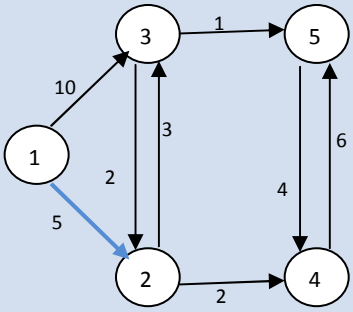


Figura 7 – Exemplo de rede

nstância	S	$\overline{S}$	$d(i)$	$ant(i)=s$	Esquema
1ª	$S=\{1\}$	$\overline{S}=\{2,3,4,5\}$	$d(1)=0$ $d(2)=5$ $d(3)=8$ $d(4)=7$ $d(5)=9$	$ant(2)=1$	

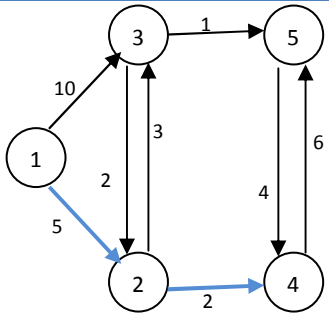
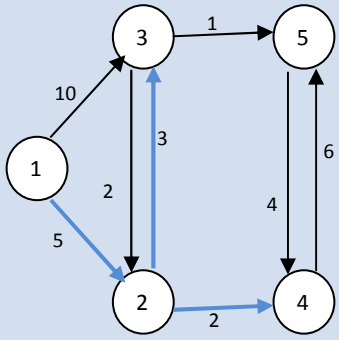
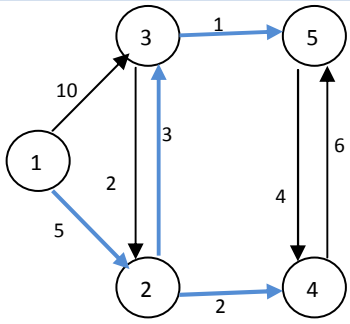
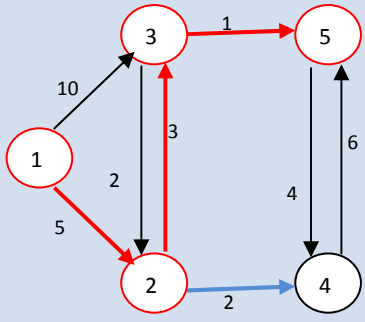
2ª	$S=\{1,2\}$	$\overline{S}=\{3,4,5\}$	$d(1)=0$ $d(2)=5$ $d(3)=8$ $d(4)=7$ $d(5)=9$	$ant(4)=2$	
3ª	$S=\{1,2,4\}$	$\overline{S}=\{3,5\}$	$d(1)=0$ $d(2)=5$ $d(3)=8$ $d(4)=7$ $d(5)=9$	$ant(3)=2$	
4ª	$S=\{1,2,3,4\}$	$\overline{S}=\{5\}$	$d(1)=0$ $d(2)=5$ $d(3)=8$ $d(4)=7$ $d(5)=9$	$ant(5)=3$	
5ª	$S=\{1,2,3,4,5\}$	$\overline{S}=\{\ \}$	$d(1)=0$ $d(2)=5$ $d(3)=8$ $d(4)=7$ $d(5)=9$	$ant(5)=3$	

Tabela 1 – Resolução, pelo algoritmo de Dijkstra, do caminho mais curto numa rede

3.3 Complexidade

Tendo em consideração duas operações básicas – a selecção de nós e as actualizações de distâncias – o algoritmo de Dijkstra é de ordem $O(n^2)$. No que concerne à selecção de nós, pode verificar-se que são percorridos n nós e, em cada um deles, analisadas todas as etiquetas temporárias, logo o número de operações envolvidas será: $n+(n-1)+(n-2)+...+1=\frac{n^2+n}{2}=O(n^2)$. Quanto à actualização de distâncias, cada uma requer tempo $O(l)$, sendo $|A(i)|=m$, pelo que, no total, é requerido tempo $O(n^2)$.

Existem várias implementações do algoritmo de Dijkstra, podendo ser consultado um resumo das diferentes implementações e respectiva complexidade em Ahuja et al. (1993, pp. 122).

3.4 Correcção de etiquetas

O algoritmo de Dijkstra resolve problemas em que as distâncias entre os nós são não negativas. Quando não é garantida a não negatividade dessas distâncias, mas não existem circuitos de peso negativo, para encontrar o caminho mais curto entre nós recorre-se ao chamado algoritmo de correcção de etiquetas. Este algoritmo baseia-se no seguinte critério de optimalidade:

Teorema: Para todos os nós $i \in V$, seja $d(i)$ a distância de um caminho do nó origem, *s*, para o nó i. Então, $d(i)$ é a distância do caminho mais curto de *s* para *i*, $\forall i \in V$, se e só se satisfaz a condição de optimalidade:

$$d(i) \le d(i) + c_{ij}, \quad \forall (i,j) \in A$$

Método da correcção de etiquetas

Input: $(V, A, C),\ s,\ t,\ n = |V|$

Output: d, ant

Inicialização:

$d(i) = +\infty, \qquad i \in V \setminus \{s\}$

$d(s) = 0$

$ant(s) = s$

Enquanto $d(j) > d(i) + c_{ij}$, *fazer:*

$d(j) = d(i) + c_{ij}$

$ant(j) = i$

Existem diversas implementações do algoritmo de correcção de etiquetas, o que pode ser consultado em Ahuja et al. (1993, pp. 155).

No caso geral, ou seja, caso possam existir circuitos com pesos negativos, o problema é NP-Completo (Garey e Johnson, 1979).

4 Descrição do Problema do Caminho mais Curto com Restrições de Capacidade

Neste capítulo é descrito o problema do caminho mais curto com uma restrição de capacidade.

Primeiramente o problema é formulado, sendo, depois, apresentadas algumas aplicações deste problema. É, ainda, analisada a sua complexidade.

Por vezes, associado a cada arco de uma rede existe um peso w_{ij} que representa o consumo de um determinado recurso, que pode ser tempo, dinheiro, combustível, etc. A esse recurso é usualmente associada uma restrição que limita o seu consumo a uma capacidade máxima, w. Um caminho P entre o nó origem s e o nó destino t diz-se admissível se verifica a restrição de recurso $\sum_{(i,j)\in P} w_{ij} \leq w$.

4.1 Formulação

Caso haja uma capacidade máxima de recurso, w, e considerando w_{ij} o consumo do recurso no arco *(i, j)* então o consumo do recurso ao longo do caminho *P* usando as variáveis de decisão x_{ij} é dado por $\sum_{(i,j)\in A} w_{ij} x_{ij}$. Assim, havendo restrição para a capacidade máxima de recurso, é necessário acrescentar um novo conjunto de restrições na formulação anterior:

$$\sum_{(i,j)\in A} w_{ij} x_{ij} \leq w \quad (6)$$

Designe-se por PCMCRC o problema do caminho mais curto com restrições de capacidade:

$$(PCMCRC) \qquad Min \quad \sum_{(i,j)\in A} c_{ij} x_{ij}$$

$$s.a:$$

$$\sum_{j\in V^+(s)} x_{sj} = 1 \qquad (1)$$

$$\sum_{i\in V^-(j)} x_{ij} = \sum_{i\in V^+(j)} x_{ji},\ j \in V \setminus \{s,t\} \qquad (2)$$

$$\sum_{j\in V^-(t)} x_{jt} = 1 \qquad (3)$$

$$\sum_{(i,j)\in A} w_{ij} x_{ij} \le w \qquad (6)$$

$$x_{ij} \in \{0,1\}, \forall (i,j) \in A \qquad (4)$$

Para garantir que não existem ciclos de peso negativo seria necessário incluir as restrições (5).

Note-se que, por vezes, a restrição (6) pode ser suficiente para garantir que não hajam ciclos, quando, por exemplo, a limitação de consumo for bastante apertada.

Um caso particular de grande interesse acontece quando $w_{ij} = 1$ para todos os arcos $(i,j) \in A$, isto é, existe um limite no número de arcos que o caminho pode ter, pois a restrição (6) toma a forma $\sum_{(i,j)\in A} x_{ij} \le w$ (6.1) e, como x_{ij} tem valor 1 caso o arco (i,j) faça parte do caminho mais curto ou 0 caso contrário, podem existir, no máximo w arcos. O problema resultante neste caso, quando a restrição (6) toma a forma (6.1), é designado por problema do caminho mais curto com restrições de cardinalidade.

Existem outro tipo de restrições adicionais, como por exemplo em problemas com janelas temporais que resultam como subproblemas de problemas de determinação de rotas de veículos (Desrochers et al., 1992). Nestes casos é estabelecido um período de tempo, com hora de início e de fim, durante o qual cada vértice pode ser visitado.

4.2 Aplicações

O PCMCRA tem inúmeras aplicações em questões reais. A resolução de vários problemas associados a questões do quotidiano, nomeadamente no seio empresarial. São apresentados, de seguida, alguns exemplos destes problemas:

- Ocorre como subproblema de problemas mais gerais, como a determinação de rotas de veículos, onde se pretende minimizar custos, distâncias, tempos, consumo de combustível, congestionamento de trânsito, etc. (Laporte et al. 1985, Avella et al. 2004).

- Em tratamentos de águas residuais e no *design*, a custo mínimo, de estruturas térmicas eficientes para construção de paredes e chãos, com características físicas adequadas (Elimam e Kohler, 1997).

- Em cartografia, por exemplo, aproximando curvas com mínimo erro. Os algoritmos são implementados com o intuito de obter melhores aproximações, minimizando o erro de aproximação de curvas, usando um número limitado de pontos intermédios. Este tipo de problemas é tratado como problema do caminho mais curto com restrições de cardinalidade, onde se limita o número de arcos (Dahl e Realfsen, 2000).

- Em redes de telecomunicações, quando se pretende garantir a qualidade do serviço, nomeadamente ao nível do atraso máximo das ligações. O atraso máximo no envio de informação depende não só do atraso nas ligações mas também do atraso de encaminhamento dos nós que atravessa. Neste caso, usualmente designado por restrição de cardinalidade, limita-se o número de arcos, fazendo, para todos os *(i, j)* do caminho, $w_{ij} = 1$. Como a probabilidade do fluxo ser afectado por uma avaria é tão maior quanto mais arcos o caminho tiver, com esta limitação de arcos garante-se, ainda, maior fiabilidade da rede. (SOUSA, 2006)

- Ainda em redes de telecomunicações, na procura de um caminho ao menor custo, para a transmissão das mensagens nas ligações, que cumpram determinados atrasos máximos e indicadores de fiabilidade previamente fixados. (XUE, 2000)

Para um estudo mais completo das aplicações poderá ser consultado o trabalho desenvolvido por Zieglemann, 2001.

Por vezes, restrições adicionais em PCMC são consideradas como objectivos, resultando, daí, o PCMC com multiobjectivos (ver Guerriero e Musmanno, 2001).

4.3 Complexidade

Como visto anteriormente, o problema do caminho mais curto sem restrições de capacidade e sem circuitos de peso total negativo pertence à classe $\mathcal{P}$. Acrescentando, por exemplo, uma restrição do tipo Saco Mochila, o problema passa a pertencer à classe $\mathcal{NP}$-difícil. Para mostrar que assim é, seguir-se-á a prova efectuada por Handler e Zang (1980), que reduz o problema da mochila ao PCMCRC.

Considere-se o clássico Problema do Saco-Mochila (*Knapsack Problem*). Dados uma mochila com capacidade w e um conjunto de n objectos, cada um com valor p_j e peso w_j, este problema consiste em decidir quais os objectos a colocar na mochila, respeitando o seu limite de capacidade e inserindo o número máximo de objectos possível. Considerando as variáveis de decisão x_j que tomam o valor 1 se o objecto é seleccionado e o valor 0 caso contrário, matematicamente este problema, *(KP)*, pode ser definido do seguinte modo:

$$
\begin{aligned}
(KP) \qquad & Max \sum_{j=1}^{n} p_j x_j \\
& s.a: \\
& \sum_{j=1}^{n} w_j x_j \leq w \\
& x_j \in \{0,1\},\ j \in \{1,...,n\}
\end{aligned}
$$

Considere-se uma rede com *n* nós e com dois arcos paralelos a sair de cada nó j para nó seguinte. Sendo $M = Max\{p_j ; j = 1,...,n-1\}$ e $c_{jj+1} = -p_j$, repare-se que se tem $M - p_j \geq 0$ para todo o *j*. Considere-se, ainda, um arco alternativo com gasto nulo que garanta sempre um caminho. Neste caso estamos perante um multigrafo, pois existem arcos paralelos. Deste modo, pode pensar-se no problema da mochila como um problema do caminho mais curto, associando as ideias: se o arco com valor e peso dados por $(M - p_i, w_i)$ é escolhido, então o objecto é colocado na mochila. Se o arco escolhido é o de valor e peso $(M,0)$, isso significa que o objecto não é colocado na mochila. A figura 9 ilustra esta situação:

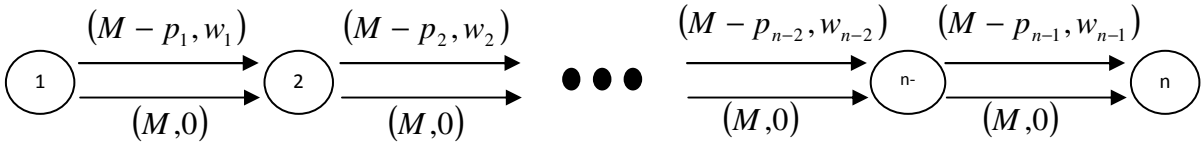


Figura 9 – Rede para o *Knapsack Problem*

Atendendo à existência de arcos paralelos, na figura 8 está representado um multigrafo, conforme já mencionado anteriormente. Para que tenhamos um PCMCRC, basta transformar o multigrafo representado na figura 8 num grafo. Essa transformação é efectuada através da duplicação dos vértices e as arestas, conforme a passagem da figura 10 para a figura 11:

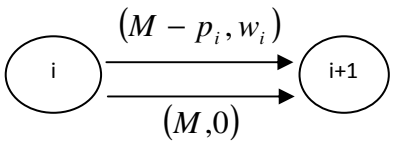


Figura 10

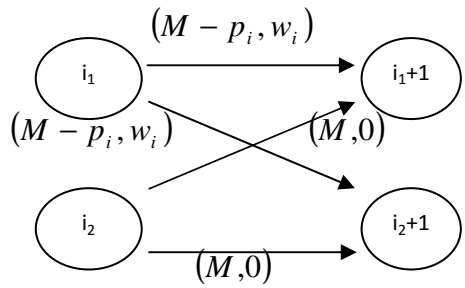


Figura 11

Sendo o problema da mochila NP-difícil, e, como se mostrou, o mesmo pode ser reduzido a um PCMCRC, então o problema do caminho mais curto com restrições é, também, NP-difícil. Este problema aparece como o problema ND30 em Garey and Johnson (1979).

5 Métodos de Resolução do PCMCRC

Neste capítulo serão abordados alguns métodos e estratégias para a resolução do problema do caminho mais curto com restrição de capacidades, não se pretendendo efectuar uma exposição exaustiva.

Dar-se-á ênfase aos algoritmos de Programação Dinâmica e aos métodos de correcção de etiquetas – que são, provavelmente, os que têm colhido mais atenção – e aos métodos baseados na relaxação da restrição de consumo de recurso do problema.

5.1 Programação Dinâmica

Uma das técnicas mais estudadas para a resolução do PCMCRC é a programação dinâmica.

A programação dinâmica é uma técnica de resolução de problemas que divide o problema original em sub-problemas de mais fácil resolução. O problema do caminho mais curto com restrições de capacidade pode ser resolvido com recurso à programação dinâmica: um caminho óptimo de s para t é construído a partir de sub-caminhos óptimos.

Seja $f_j(r)$ o custo do menor caminho com peso menor ou igual a r do nó origem para o nó j.

Fazendo $f_1(r) = 0, \forall r \in \{0,...,w\}$ e $f_j(0) = +\infty, \forall j \in \{2,...,n\}$, obtém-se a seguinte equação recursiva:

$$f_j(r) = \min\left\{ f_j(r-1), \min_{\substack{(i,j)\in A \\ w_{ij}\le r}} \{ f_j(r-w_{ij}) + c_{ij} \} \right\}$$

O PCMCRC pode, então, ser resolvido pelo seguinte algoritmo de programação dinâmica:

Programação Dinâmica para o Problema do Caminho Mais Curto com Restrições de Capacidade

---

Input: (V, A, C), $s=0$, $V=\{0,...,n\}$

Output: $f_j(r)$, $ant_j(r)$

Inicialização:

Para $r=0$ *até* $r=w$ *fazer*

$f_0(r)=0$

$ant_0(r)=0$

Para j=2 até j=n fazer

$f_{j-1}(0)=+\infty$

$ant_{j-1}(0)=0$

Para $r=1$ *até* $r=w$ *fazer*

Para j=1 até j=n fazer

$f_{j-1}(r)=f_{j-1}(r)$

$ant_{j-1}(r)=ant_{j-1}(r-1)$

Para i=1 até i=n, $\forall i \in \{i:(i,j)\in A\}$ *fazer*

Se $w_{(i-1)(j-1)} \le r$ *e* $f_{j-1}(r-w_{(i-1)(j-1)})+c_{(i-1)(j-1)} < f_{j-1}(r)$ *fazer*

$f_{j-1}(r)=f_{j-1}(r-w_{(i-1)(j-1)})+c_{(i-1)(j-1)}$

$ant_{j-1}(r)=i$

De seguida é apresentado um exemplo de implementação deste algoritmo.

5.1.1 Exemplo

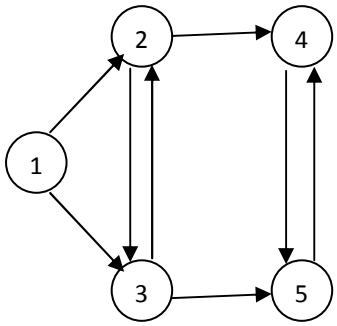


Figura 12 – Exemplo de rede

Considere-se a figura 12, onde é apresentada uma rede cuja matriz de adjacência A, a matriz dos custos C e a matriz dos pesos W são apresentadas de seguida. Seja, ainda, w=10.

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \qquad C = \begin{bmatrix} 0 & 5 & 10 & +\infty & +\infty \\ +\infty & 0 & 3 & 2 & +\infty \\ +\infty & 2 & 0 & +\infty & 1 \\ +\infty & +\infty & +\infty & 0 & 6 \\ +\infty & +\infty & +\infty & 4 & 0 \end{bmatrix}$$

$$W = \begin{bmatrix} 0 & 1 & 2 & +\infty & +\infty \\ +\infty & 0 & 1 & 2 & +\infty \\ +\infty & 3 & 0 & +\infty & 4 \\ +\infty & +\infty & +\infty & 0 & 1 \\ +\infty & +\infty & +\infty & 3 & 0 \end{bmatrix}$$

	$f_j(r)$				
j / ***r***	***1***	***2***	***3***	***4***	***5***
0	0	5	$+\infty$	$+\infty$	$+\infty$
1	0	5	8	$+\infty$	$+\infty$
2	0	5	8	7	$+\infty$
3	0	5	8	7	13
4	0	5	8	7	13
5	0	5	8	7	9
6	0	5	8	7	9
7	0	5	8	7	9
8	0	5	8	7	9
9	0	5	8	7	9
10	0	5	$+\infty$	$+\infty$	$+\infty$

Tabela 2 - Tabela de custos

	Antecessores				
j / ***r***	***1***	***2***	***3***	***4***	***5***
0	0	0	0	0	0
1	0	1	0	0	0
2	0	1	2	0	0
3	0	1	2	2	0
4	0	1	2	2	4
5	0	1	2	2	4
6	0	1	2	2	3
7	0	1	2	2	3
8	0	1	2	2	3
9	0	1	2	2	3
10	0	1	2	2	3

Tabela 3 - Tabela de antecessores

É possível estabelecer as equações de programação dinâmica considerando como condições iniciais as respeitantes ao nó *n*, isto é, considerando os sub-caminhos de j para n.

De seguida é reformulado o PCMCRC como um PCMC. Para isso recorre-se a um grafo expandido – ver figura 13 – cujos nós são os estados da programação dinâmica e os arcos representam as possíveis transições entre estados.

Repare-se, na figura 13, que muitos estados podem ser eliminados, por existir um caminho mais curto com um gasto inferior. É o exemplo dos estados (1, k), com k=1,…,10, bem como todos os que apenas se obtêm a partir desses, tais como os (2, k), com k=2,…,10. Estes estados podem ser eliminados porque existe a ligação do vértice 1 a custo zero para o vértice 2 a custo 1. Pode ainda observar-se que há medida que o vértice índice *i* se aproxima do número total de vértices, *n*, o número de estados dominados vai diminuindo, existindo mais estados dominados para $i = 1{,}2$ do que para $i = 4{,}5$, por exemplo.

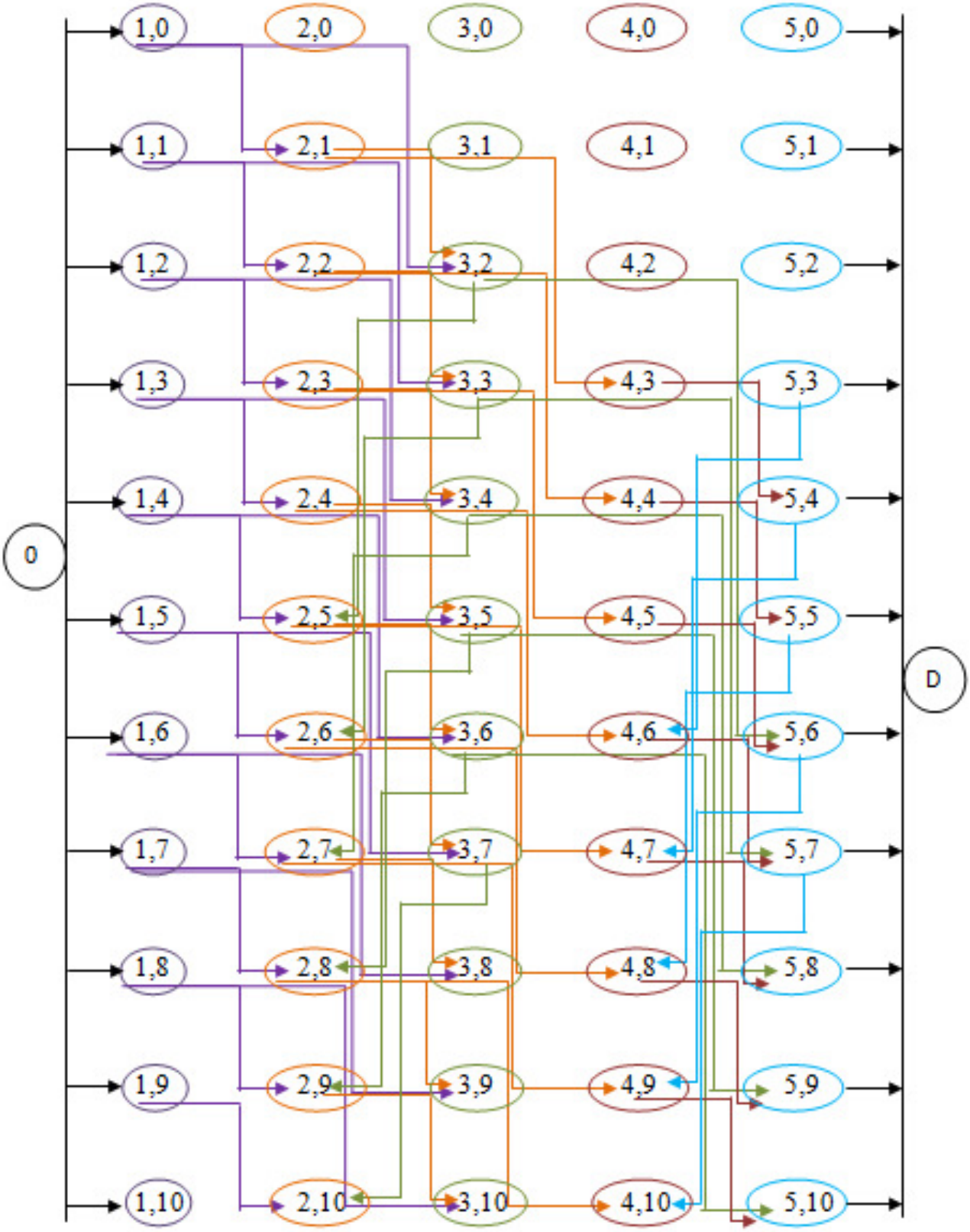


Figura 13 – Formulação do PCMCRC como PCMC

5.1.2 Fixação de etiquetas

A metodologia da fixação de etiquetas é vastamente utilizada como melhoramento dos algoritmos de programação dinâmica que resolvem o PCMCRC. Basicamente, estes métodos destinam-se a gerar, de uma forma eficiente, apenas os caminhos não dominados. De modo a que isso aconteça será necessário proceder-se à eliminação de estados dominados, tal como exemplificado no Exemplo 5.1.1..

Giovanni Righini e Matteo Salani (2006) descrevem um algoritmo que utiliza esta metodologia. Em cada estado, que significa um caminho de *s* para *i*, é considerado *R*, o vector dos consumos de recurso, e *C*, o custo associado ao estado. Estados diferentes para um mesmo vértice correspondem a diferentes caminhos possíveis (soluções admissíveis) que envolvem esse vértice.

Alguns espaços de estados poderão, nalgumas situações, ser eliminados. Por exemplo, para um dado vértice i, um estado t_1 pode ser eliminado se existe um estado t_2 tal que $R(t_1) \geq R(t_2)$ e $C(t_1) \geq C(t_2)$.

O procedimento consiste em efectuar, em cada estado, testes de dominância, gravando apenas os estados não dominados.

Também Xiaoyan Zhu (2005) propôs uma técnica de redução de estados baseada na eliminação de arcos e nós. A eliminação de arcos é efectuada com base na limitação de recursos, sem que seja tido em conta qualquer informação relativamente ao custo associado a cada arco.

Tendo em consideração que o número de estados vai aumentando, alguns autores propuseram métodos de pesquisa bidireccional, iniciando a pesquisa pelo primeiro e pelo último nó (RIGINI e SALANI, 2006).

5.2 Relaxações

Seja *P* um Problema de Programação Inteira. Obtém-se uma relaxação de *P*, *RP*, aumentando o conjunto de soluções admissíveis e/ou trabalhando com outra função objectivo que tenha um valor igual ou superior (inferior) para os problemas de maximização (minimização) em toda a solução admissível (Wolsey, 1998l, pp. 24).

Verifica-se que caso uma relaxação não tenha soluções admissíveis, então o problema original também não tem.

5.2.1 Relaxação Linear

Seja *P* um Problema de Programação Inteira em que o conjunto das restrições é definido por $x \in X \cap Z^n$. Uma relaxação linear de *P*, *RL*, obtém-se aumentando o conjunto de restrições, fazendo $x \in X$.

No caso concreto do PCMCRC, a relaxação linear obtém-se substituindo na formulação a restrição $x_{ij} \in \{0,1\}$ por $x_{ij} \in [0,1]$.

Zieglemann (2001) propôs uma formulação diferente do Problema do Caminho mais Curto com restrições adicionais:

Para todo o caminho *p*, com $p \in P$, onde *P* é o conjunto dos índices dos caminhos, de *s* para *t* introduz-se uma variável binária que tome valores 1 ou 0, x_p, consoante seja ou não seleccionado o caminho, respectivamente, e usa-se c_p e $r^{(i)}_p$ ($i=1,\ldots,k$) para designar o custo e o consumo de recurso de *p*, respectivamente.

$$Min \quad \sum_p c_p x_p \qquad (7)$$

$$s.a:$$

$$\sum_p x_p = 1 \qquad (8)$$

$$\sum_p r_p x_p \le \lambda \qquad (9)$$

$$x_p \in \{0,1\}, p \in P \qquad (10)$$

Repare-se que (8) garante que x_p =1 apenas num caso, isto é, é seleccionado um único caminho, (9) garante que o caminho obedece à restrição de limitação do consumo e (7) garante o caminho com menor custo.

Relaxando a restrição (10), verifica-se que o valor da relaxação do problema é um limite inferior para o valor da função objectivo.

O dual deste problema, para o caso de uma restrição de limitação de recursos, tem duas variáveis, u e $v \leq 0$:

$$Max \quad u + \lambda v \qquad (11)$$

$$s.a:$$

$$u + r_p v \leq c_p, \forall p \in P \qquad (12)$$

$$v \leq 0 \qquad (13)$$

Este problema pode ser interpretado geometricamente, considerando que cada par (u,v) é um ponto do plano u,v e cada restrição define um semiplano em associação à direcção $(1,\lambda)$:

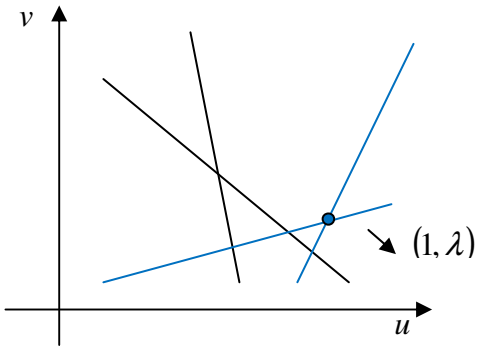


Figura 14 – Plano u,v: procurar o óptimo na direcção $(1,\lambda)$

No plano r,c, cada ponto (u,v) do plano u,v é interpretado como uma recta $c = vr + u$. Assim, cada restrição $u + r_p v \leq c_p$ é interpretada como um ponto (r_p, c_p) do plano r,c. Procura-se, então, uma recta com declive negativo que maximize $u + \lambda v$, de modo a que tenha o máximo valor para $r = \lambda$ e que todos os pontos (r_p, c_p) estejam acima dela:

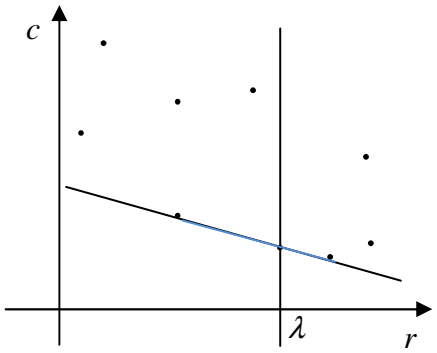


Figura 15 – Plano r,c: procurar a recta que tenha máximo valor para $r = \lambda$ e que todos os pontos (r_p, c_p) estejam acima dela

Com base nesta interpretação geométrica Zieglemann (2001) propõe um algoritmo combinatório denominado por "The hull approach" para resolver este problema dual.

5.2.2 Relaxação Lagrangeana

Existem várias referências a algoritmos propostos para resolver o PCMCRC, relaxando a restrição de capacidade.

Minoux e Ribeiro (1985,1986) consideraram o caso do problema do caminho mais curto quando existe limitação superior e inferior de recursos. Com a finalidade de encontrar um limite inferior para a solução do problema, Handler e Zang (1980) relaxaram a restrição dos recursos à moda lagrangeana e, resolvendo o dual lagrangeano, maximizaram o limite inferior. Notando que um algoritmo para resolver o problema consiste sem determinar os k caminhos mais curtos, onde k representa o primeiro caminho admissível, e que esse algoritmo é impraticável quando k é elevado, estes autores recorreram à relaxação lagrangeana de modo a reduzirem o valor de k.

De seguida vamos apresentar a relaxação lagrangeana obtida por relaxação das restrições de recurso.

De volta à formulação do problema do caminho mais curto:

$$(PCMCRC) \qquad Min \sum_{(i,j)\in A} c_{ij} x_{ij}$$

$$s.a:$$

$$(1), \quad (2), \quad (3), \quad (4), \quad (6)$$

Relaxando as restrições associadas aos recursos consumidos, (6), ou seja, associando-lhe um multiplicador $\lambda > 0$ e inserindo-a na função objectivo, obtém-se o problema relaxado:

$$L(\lambda) = Min \sum_{(i,j)\in A} c_{ij} x_{ij} - \lambda\left(w - \sum_{(i,j)\in A} w_{ij} x_{ij} \right)$$

$$s.a:$$

$$(1), \quad (2), \quad (3), \quad (4)$$

A função objectivo pode escrever-se da forma: $\sum_{(i,j)\in A}(c_{ij}+\lambda w_{ij})x_{ij}-\lambda w$.

A restrição (6) tem associada um multiplicador de Lagrange e a função objectivo sofre uma penalização pela violação da restrição.

Seja $L(\lambda)$ a função lagrangeana do PCMCRC.

Lema: Para qualquer $\lambda>0$ de multiplicadores de Lagrange, $L(\lambda)$ é um limite inferior na função objectivo do problema original.

Para obter o melhor limite inferior é necessário resolver o problema de optimização $L^*=\max_{\lambda} L(\lambda)$, designado por problema do multiplicador de Lagrange, ou dual lagrangeano associado ao problema original.

Teorema (Dualidade): $L^*\le z^*$.

No PCMCRC, o óptimo do dual lagrangeano é igual ao da relaxação linear do Problema Linear Inteiro associado, porque a formulação do caminho mais curto correspondente ao problema relaxado é inteira, isto é, $L(\lambda)$ não se altera considerando a respectiva relaxação linear.

Assim, neste caso concreto, a relaxação lagrangeana surge como um método alternativo para a resolução da relaxação linear deste problema.

5.2.3 Penalidades

Considere-se a formulação do PCMCRC com várias restrições de capacidade:

$$(PCMCRC) \qquad Min \quad \sum_{(i,j)\in A} c_{ij}x_{ij}$$

$$s.a:$$

$$(1),\quad (2),\quad (3),\quad (4),\quad (14)$$

Sendo $\sum_{(i,j)\in A} w^k_{ij\,ij}x_{ij}\le w^k, k=1,...,K \quad (14)$.

Designado as restrições (14) por "difíceis" e as restantes por "fáceis", Avella et al. (2004) propõe uma relaxação das restrições difíceis por uma função exponencial de penalidade, $\phi(x) = e^{\alpha\left(\frac{\sum w_{ij}^1 x_{ij}}{w^1} - 1\right)} + ... + e^{\alpha\left(\frac{\sum w_{ij}^K x_{ij}}{w^K} - 1\right)}$, com $\alpha > 0$ um parâmetro de calibração.

Esta função, $\phi(x)$, "substitui" a medida de admissibilidade da solução x:

- Se x é admissível, as restrições difíceis são satisfeitas e, portanto, $\phi(x)$ não é maior que 1;
- Se $\phi(x)$ é maior que 1, então x não é admissível, violando as restrições difíceis.

Deste modo, o problema consiste em encontrar uma solução x^* tal que $\phi(x^*) \leq 1$. Define-se, então, um subproblema da admissibilidade que consiste em minimizar a função de penalidade $\phi(x)$, sujeita às restrições fáceis do problema.

De modo a garantir a qualidade (em termos do valor da função objectivo) das soluções admissíveis, e tendo em conta um valor inicial de uma solução admissível, C_0, é incluída, nas restrições difíceis, a restrição $\sum_{(i,j)\in A} c_{ij} x_{ij} \leq C_0$.

No nosso problema, com uma única restrição de recurso, $\sum_{(i,j)\in A} w_{ij} x_{ij} \leq w$, a função penalidade é definida do seguinte modo: $\phi(x) = e^{\alpha\left(\frac{\sum w_{ij} x_{ij}}{w} - 1\right)} + e^{\alpha\left(\frac{\sum c_{ij} x_{ij}}{C_0} - 1\right)}$.

O subproblema (da admissibilidade) acima referido resolve-se utilizando-se um algoritmo que começa com uma solução x^0 e gera uma sequência de soluções $x^{h+1} = x^h + \mu^h d^h$ convergentes para uma solução $\varepsilon-$admissível, que é uma solução que verifica $\sum w_{ij} x_{ij} \leq (1+\varepsilon) w$ e $\sum c_{ij} x_{ij} \leq (1+\varepsilon) C_0$. Assim, este algoritmo consiste, num primeiro passo em encontrar uma direcção admissível e, num segundo passo, calcular o tamanho do passo.

A direcção admissível, d^h, é encontrada resolvendo o seguinte problema linear, onde P é o poliedro definido pelas restrições "fáceis" que no nosso caso são as (1), (2), (3) e (4). Considerando os vectores linha $W = (w_{ij})_{(i,j)\in A}$ e $C = (c_{ij})_{(i,j)\in A}$, e as funções

$\phi_1(x) = e^{\alpha\left(\frac{\sum w_{ij}^1 x_{ij}}{w}\right)}$ e $\phi_2(x) = e^{\alpha\left(\frac{\sum c_{ij}^1 x_{ij}}{C_0}\right)}$, tem-se: $\nabla\phi(x^h) = \left(\frac{\alpha}{w} W\phi_1(x^h) + \frac{\alpha}{C_0} C\phi_2(x^h)\right)$, e

$$\begin{aligned} &\min \quad (\nabla\phi(x^h))d \\ &s.a: \quad x^h + d \in P \end{aligned}$$

Fazendo $x^h + d = y$, o problema pode ser escrito da forma seguinte:

$$\begin{aligned} &\min \quad (\nabla\phi(x^h))y \\ &s.a: \quad y \in P \end{aligned}$$

Este é o PCMC com a função objectivo alterada. Repare-se que, neste caso, o coeficiente associado a cada arco *(i, j)* é uma soma ponderada dos coeficientes c_{ij} e w_{ij}.

Considerando y^h um vértice do poliedro P, isto é, um caminho que pode não verificar a restrição (6), o tamanho (dinâmico) do passo é determinado resolvendo o problema de minimização que se segue:

$$\begin{aligned} &\min \quad \phi(\mu x^h + (1-\mu) y^h) \\ &0 \le \mu \le 1 \end{aligned}$$

Seja μ^h a solução óptima deste problema. O novo ponto interno do segmento $[x^h, y^h]$, x^{h+1}, é dado por: $x^{h+1} = \mu^h x^h + (1-\mu^h) y^h$.

Este processo decorre até que o número máximo, predefinido, de iterações é atingido ou, alternativamente, é encontrada uma solução admissível. Neste caso actualiza-se y^h, z e C_0, repetindo-se, novamente, o processo.

Prova-se que, com uma escolha adequada do parâmetro de calibração, α, da função $\phi(x)$, o método converge em tempo polinomial.

Vários autores já se debruçaram sobre a escolha deste parâmetro. Plotkin e Karger (1995) provaram que caso exista uma solução admissível, o algoritmo converge em tempo polinomial para uma solução admissível fazendo $\alpha = \log\frac{3m}{\varepsilon}$. Experiências computacionais mostraram que este valor é muito elevado. Bienstock (1998) considerou $\alpha = \frac{0{,}2}{\varepsilon}$. Avella et al. (2004) consideraram $\alpha \in [10{,}100]$.

Deste algoritmo resulta um caminho admissível, que satisfaz a restrição difícil.

De modo a que os mesmos caminhos não sejam gerados mais do que uma vez, é adoptada uma lista na qual se registam as soluções admissíveis já obtidas. Esta lista é verificada aquando do início do processo, sendo designada por lista Taboo. Neste caso, quando um caminho é ignorado, escolhe-se o próximo caminho mais curto que não esteja na lista Taboo. Como Avella et al. (2004) notaram, esta alteração faz com que o algoritmo deixe de ter garantia de convergência para uma solução ε-admissível.

6 Heurística de aproximação

Neste capítulo é descrito um algoritmo heurístico para a resolução do PCMCRC baseado na relaxação da restrição de capacidade, designado por Algoritmo B. Este algoritmo parte de um caminho inicial e determina sucessivamente vários caminhos até obter uma solução admissível para o problema, isto é, um caminho mais curto que respeite a restrição de capacidade.

O algoritmo baseia-se na ideia de relaxar a restrição de recurso, incluindo, de algum modo, na função objectivo a informação da restrição relaxada. Esta ideia está já presente na relaxação lagrangeana e no método das penalidades, descritos no capítulo 5, mas este novo algoritmo pretende incluir a informação da restrição de uma forma diferente: penalizando apenas os arcos no caminho caso este não seja admissível.

Este algoritmo, o Algoritmo B, é baseado num Algoritmo A que, considerando custos c'_{ij}, começa por procurar o caminho mais curto do nó origem para o nó destino sem ter em conta a restrição de capacidade, averiguando, de seguida, se o caminho encontrado é admissível. Caso não seja, o Algoritmo A penaliza os arcos desse caminho aumentando os custos dos seus arcos.

O Algoritmo B consiste em correr o Algoritmo A, com $c'_{ij} = c_{ij}$ e, caso não se encontre uma solução admissível ao fim de um número fixado de iterações, a função objectivo é alterada. Mais propriamente os custos são alterados, fazendo-se $c'_{ij} = \mu \times c_{ij} + (1-\mu) \times w_{ij}$, onde $\mu = 1$ no início, sendo actualizado (decrementado até ao limite 0) em cada iteração. Assim, num extremo, este algoritmo trata o problema sem restrições (tendo em conta, apenas, os coeficientes c_{ij}) – caso em que $\mu = 1$ – e no outro extremo a função objectivo tem em conta, apenas, os coeficientes w_{ij} da restrição – caso em que $\mu = 0$.

O Algoritmo A e o Algoritmo B são, de seguida, descritos com maior detalhe.

6.1 Algoritmo A

O Algoritmo A começa por procurar o caminho mais curto, *P*, do nó origem para o nó destino sem ter em conta a restrição de capacidade.

De seguida, averigua se o caminho encontrado é admissível, isto é, se verifica a restrição de capacidade. Se assim for, o algoritmo pára, tendo sido encontrada uma solução admissível.

Caso contrário, se o caminho, *P*, não é admissível, o algoritmo penaliza-o. Essa penalização consiste em adicionar aos custos dos arcos do caminho, *P*, um incremento positivo, Δ.

O incremento Δ é escolhido de tal forma a que o critério de optimalidade (ver secção 3.4) deixe de se verificar, ou seja, de modo a que $d(i)$ deixe de ser o caminho mais curto do nó origem até ao nó *i* para algum nó *i* no caminho *P*. Para que tal aconteça procede-se da seguinte forma:

- Para cada nó *i* do caminho não admissível encontrado, exceptuando o nó origem, determina-se Δ_i, considerando-se, assim, os antecessores t do nó *i* que não sejam antecessores de *i* no caminho (não admissível) encontrado e determina-se, para cada t, $\Delta_i = \min_{\substack{t \in V^-(i) \\ t \neq ant(i)}} \{d(t) + c_{ti} - d(i)\}$.
- Desses Δ_i's, escolhe-se o menor, adicionando-lhe, ainda, uma unidade para garantir a destruição do critério, ou seja, $\Delta = \min_{\substack{i \in P \\ i>1}} \{\Delta_i\} + 1$.

O algoritmo prossegue, voltando a procurar o caminho mais curto do nó origem para o nó destino (após a actualização dos custos pelo incremento Δ), procedendo-se de igual modo até que o caminho encontrado seja admissível ou o número máximo de iterações seja atingido.

De uma forma muito simples, o Algoritmo A pode ser descrito da seguinte forma:

Passo 1: Gerar caminho mais curto

Passo 2: Se o caminho é admissível ou o número máximo de iterações for atingido

PARAR

Senão:

somar $\Delta > 0$ *ao custo de todos os arcos do caminho e voltar ao Passo 1*

Passar-se-á à descrição do Algoritmo A em pseudo-código.

Neste algoritmo consideram-se:

- CMC(V,A,C'): o algoritmo do caminho mais curto descrito na secção **3.1.2.**.
- P: o caminho resultante da aplicação do algoritmo CMC(V,A,C').
- $d(i)$: distância do caminho mais curto do nó origem, s , para o nó i .
- $ant(i)$: antecessor de *i* no caminho mais curto do vértice *s* para o vértice *i.*
- *adm*: variável booleana que indica se o caminho encontrado verifica, ou não, a restrição de capacidade.
- n_{iter}: contador do número de iterações
- max_{iter}: número máximo de iterações

Algoritmo A

Input: (V, A, C´), n_{iter}, max_{iter}

Output: P, adm

Inicialização:

$c'_{ij} = c_{ij}$

$\Delta = 0$

adm =0

$n_{iter}=0$

Enquanto (adm=0 e $n_{iter} \leq max_{iter}$*) fazer:*

CMC(V, A, C´)

fazer $x_{ij} = 1$ *se (i, j) está no caminho P e* $x_{ij} = 0$ *caso contrário.*

Se $\sum x_{ij} w_{ij} \leq w$ *então*

adm=1

PARAR (O caminho P encontrado é admissível)

Caso contrário fazer

$$\Delta_j = +\infty, \quad \forall j \in V$$

Para todo $(i \in P \quad e \quad i > 1)$ *fazer*

$$\Delta_i = \min_{\substack{t \in V^-(i) \\ t \neq ant(i)}} \{d(t) + c_{ti} - d(i)\}$$

$$\Delta = \min_{\substack{i \in P \\ i>1}} \{\Delta_i\} + 1$$

$$c'_{ij} = c'_{ij} + \Delta, \quad \forall i, j \in P$$

6.1.1 Exemplo

Observe-se a rede e as matrizes de pesos e custos do exemplo 5.1.1.. Considere-se, agora, W=5, ou seja, o PCMCRC tem, como restrição, o consumo limitado a 5.

Na tabela 4 está resolvido esse PCMCRC. Para a 1ª e para a 2ª instâncias, na coluna correspondente ao caminho mais curto da tabela 4, cada nó tem indexada a sua marca, $d(i)$, para facilitar a resolução, bem como a leitura, do exemplo. Observe-se, então, o funcionamento do Algoritmo A neste caso concreto:

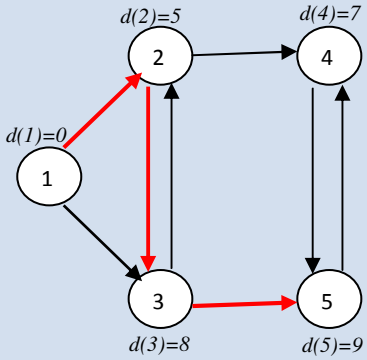
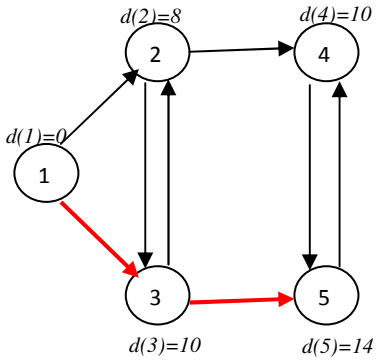
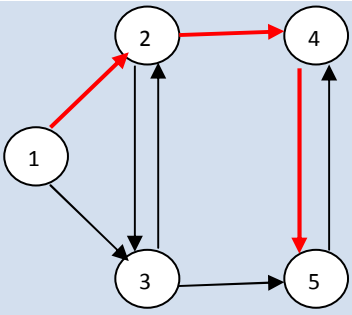
Instância	Caminho mais curto	adm	Δ_j	Δ	C'
1ª		0	$\Delta_2 = d(3) + c_{3,2} - d(2) = = 8 + 2 - 5 = 5$ $\Delta_3 = d(1) + c_{1,3} - d(3) = = 0 + 10 - 8 = 2$ $\Delta_5 = d(4) + c_{4,5} - d(5) = = 7 + 6 - 9 = 4$	Δ=2+1= =3	$\begin{bmatrix} 0 & 8 & 10 & +\infty & +\infty \\ +\infty & 0 & 6 & 2 & +\infty \\ +\infty & 2 & 0 & +\infty & 4 \\ +\infty & +\infty & +\infty & 0 & 6 \\ +\infty & +\infty & +\infty & 4 & 0 \end{bmatrix}$
2ª		0	$\Delta_3 = d(2) + c_{2,3} - d(3) = = 8 + 6 - 10 = 4$ $\Delta_5 = d(4) + c_{4,5} - d(5) = = 10 + 6 - 14 = 2$	Δ=2+1= =3	$\begin{bmatrix} 0 & 7 & 13 & +\infty & +\infty \\ +\infty & 0 & 5 & 2 & +\infty \\ +\infty & 2 & 0 & +\infty & 7 \\ +\infty & +\infty & +\infty & 0 & 6 \\ +\infty & +\infty & +\infty & 4 & 0 \end{bmatrix}$
3ª		1			

Tabela 4 – Resolução de um PCMCRC usando o Algoritmo A.

Repare-se que, ao fim de 3 iterações o algoritmo convergiu, tendo sido encontrada uma solução admissível: o caminho 1-2-4-5.

Como se poderá observar no exemplo que de seguida se apresenta, não há garantia que o Algoritmo A convirja para uma solução admissível, sendo, então, necessário apresentar algumas alternativas.

6.1.2 Exemplo

Considere-se a rede seguinte, onde os pares ordenados sob os arcos indicam, respectivamente, o custo e o peso, ou seja, tem-se, em cada arco (i, j) em A, (c_{ij}, w_{ij}). Neste caso, W=3 é o limite de consumo de recurso, e pretende-se o caminho mais curto, sujeito à restrição de limitação de consumo de recurso, do nó 1 para o nó 4.

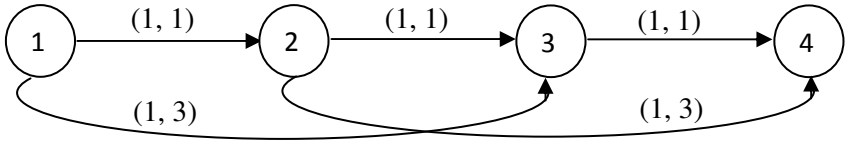


Figura 16

Repare-se que o caminho mais curto sem restrição de capacidade é o caminho 1-3-4, por exemplo. Penalizando esse caminho, ou seja, adicionando $\Delta_1 = 1$ aos custos que lhe estão associados, vem:

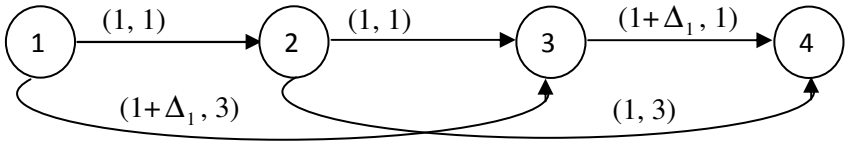


Figura 17

O algoritmo prosseguirá, escolhendo o caminho 1-2-4. Penalizando esse caminho, ou seja, adicionando $\Delta_2 = 3$ aos custos que lhe estão associados, vem:

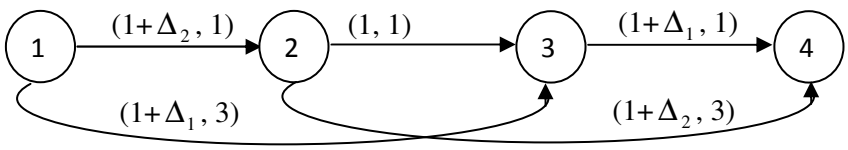


Figura 18

O algoritmo escolherá, então, o caminho 1-3-4, e assim sucessivamente. Neste caso, verifica-se que o algoritmo entra em ciclo, usando alternadamente os caminhos 1-2-4 e 1-3-4, nunca gerando o único caminho admissível, que é o 1-2-3-4.

Podem ser estudadas algumas alternativas para evitar que tal aconteça. Uma delas será verificar quais os arcos mais repetidos ao longo das iterações e penalizá-los ainda mais ou mesmo removê-los.

Outra opção é, tal como apresentado por Avella et al. (2004), adoptar uma lista com todos os caminhos já gerados, ou seja, uma lista Taboo, e gerar os k caminhos mais curtos, sendo que k é o índice do primeiro caminho gerado que não consta da lista Taboo.

Uma terceira alternativa, a que será ser utilizada no Algoritmo B, passa por penalizar todos os arcos proporcionalmente ao seu peso.

6.2 Algoritmo B

Considera-se uma função objectivo que, num extremo não tem em conta a restrição de capacidade, mas apenas os coeficientes c_{ij} e, noutro extremo, apenas tem em conta os coeficientes da restrição, w_{ij}. A função objectivo terá, então, a forma $\sum_{(i,j)\in A} c'_{ij}\, x_{ij} = \sum_{(i,j)\in A} \mu c_{ij} x_{ij} + (1-\mu) w_{ij} x_{ij}$, com $\mu \in [0,1]$. A inicialização é feita com $\mu = 1$, sendo este valor actualizado, em cada iteração, segundo um *passo*, tendo-se escolhido *passo=0,1*, efectuando-se, então, $c'_{ij} = \mu \times c_{ij} + (1-\mu) \times w_{ij}$. Depois é corrido o algoritmo A e, no final, actualizado o μ da seguinte forma: $\mu = \mu$ - *passo*. Este procedimento é efectuado enquanto μ for positivo.

Esta alternativa poderá ser acrescentada ao algoritmo A, surgindo um novo algoritmo, o Algoritmo B, cuja descrição em pseudo-código é efectuada de seguida.

Algoritmo B

Input: (V, A, C´), n_{iter}, max_{iter}

Output: P, adm,

Inicialização:

$\mu = 1$

passo=0,1

adm=0

Enquanto (adm=0 e $\mu \geq 0$ *) fazer:*

$c'_{ij} = c_{ij} + \mu \times c_{ij} + (1 - \mu) \times w_{ij}$

Algoritmo A(C', adm)

$\mu = \mu$ - *passo*

Note-se que quando $\mu = 0$ se tem $c'_{ij} = 0 \times c_{ij} + (1-0) \times w_{ij} = w_{ij}$. Neste caso garante-se a devolução de uma solução admissível, caso exista.

7 Estudo Computacional

Neste capítulo será efectuado um estudo computacional. Seguindo o estudo de Pisinger (2005) para o problema do Saco-Mochila tentar-se-á compreender quais as instâncias mais difíceis do PCMCRC. Ao mesmo tempo será estudado, computacionalmente, o desempenho do Algoritmo B.

Neste estudo são considerados os seguintes algoritmos para a resolução do PCMCRC:

- Branch & Bound;
- Relaxação Linear (RL);
- Programação Dinâmica (PD);
- Algoritmo B.

Após a escrita em linguagem Mosel dos algoritmos supra mencionados, geraram-se várias instâncias com algumas combinações dos parâmetros de entrada.

Nos casos em que a programação dinâmica foi muito demorada, optou-se por não a incluir nos testes.

Verificou-se que para instâncias onde existe correlação positiva e forte entre os custos e os pesos, caso estudado, também, por Pisinger (2005) para o problema do Saco-Mochila, o caminho mais curto sem restrição de capacidade coincide com o caminho mais curto com restrição de capacidade; pois, neste caso, o caminho mais curto coincide, usualmente, com o caminho de menor peso. Deste modo, não serão apresentados os resultados dessas instâncias, por se poderem considerar triviais.

Numa fase inicial realizaram-se testes gerando aleatoriamente custos e pesos, seguindo de perto o estudo de Pisinger (2005) para o problema do Saco-Mochila, para pequenas dimensões. Nesses casos verificou-se que os problemas resultantes eram triviais, ocorrendo a situação descrita anteriormente.

De modo a ser possível encontrar instâncias difíceis, houve necessidade de se recorrer a várias tipologias de grafos. Assim, o estudo computacional divide-se em três

tipos – Tipo I, Tipo II e Tipo III. Para cada tipo de grafo houve a necessidade de criar grupos diferentes, para evitar a geração de instâncias triviais.

O Tipo I considera as instâncias em que o grafo foi estruturado por níveis. Nesta tipologia consideram-se vinte níveis, existindo apenas arcos de um nível para o nível seguinte. A origem encontra-se no primeiro nível e o destino no último nível. Estes dois níveis (primeiro e último) apenas têm estes nós (nó origem e nó destino, respectivamente). O Tipo I encontra-se organizado em três grupos: GI, GII e GIII.

O Tipo II tem em conta as instâncias geradas quando pesos e custos são gerados aleatoriamente num intervalo, tendo em consideração a distância entre os índices dos nós, e encontra-se organizado em dois grupos: GI e GII.

Por último, o Tipo III engloba as instâncias em que os pesos e os custos são gerados tendo em conta a distância euclidiana entre os nós, sendo apenas considerado um grupo: GI.

Para cada um dos tipos acima referidos foram considerados os seguintes grupos, onde *R* é um parâmetro controlado:

Para o Tipo I:

- GI, onde:
  - ✓ os pesos, w_{ij}, são gerados aleatoriamente no intervalo $[w_i, w_f]$, caso *i* e *j* pertençam a níveis consecutivos, sendo infinito caso contrário;
  - ✓ os custos, c_{ij}, são obtidos por $c_{ij} = w_f - w_{ij} + l$, sendo l gerado aleatoriamente no intervalo $[0, R \times 0{,}01]$, caso *i* e *j* pertençam a níveis consecutivos, sendo infinito caso contrário.
- GII, onde:
  - ✓ os pesos, w_{ij}, são gerados aleatoriamente no intervalo $[w_i, w_f]$, caso *i* e *j* pertençam a níveis consecutivos, sendo infinito caso contrário;
  - ✓ os custos, c_{ij}, são obtidos por $c_{ij} = l_1 + l_2$, onde l_1 é gerado aleatoriamente no intervalo $[0, w_{ij}]$ e l_2 é gerado aleatoriamente no intervalo $[0, R \times 0{,}01]$, caso i e j pertençam a níveis consecutivos, sendo infinito caso contrário.

- GIII, onde:
  - ✓ os pesos, w_{ij}, são gerados aleatoriamente no intervalo $[w_i, w_f]$, caso *i* e *j* pertençam a níveis consecutivos, sendo infinito caso contrário;
  - ✓ os custos, c_{ij}, são gerados aleatoriamente no intervalo$[w_i, w_f]$, caso *i* e *j* pertençam a níveis consecutivos, sendo infinito caso contrário.

Para o Tipo II:

- GI, onde:
  - ✓ os pesos, w_{ij}, são obtidos por $w_{ij} = w_i + l$, sendo l aleatoriamente gerado no intervalo $[0, (w_i - w_f) \times | j - i |]$;
  - ✓ os custos, c_{ij}, são obtidos por $c_{ij} = w_i \times | j - i | - w_{ij} + \frac{R}{10}$.
- GII, onde:
  - ✓ os pesos, w_{ij}, são obtidos por $w_{ij} = w_i + l$, sendo l aleatoriamente gerado no intervalo $[0, (w_i - w_f) \times | j - i |]$;
  - ✓ os custos, c_{ij}, são obtidos por $c_{ij} = l \times | j - i |$, sendo l aleatoriamente gerado no intervalo $[w_i, w_f]$.

Para o Tipo III:

- GI, apenas, onde:
  - ✓ a cada vértice *i* se faz corresponder um ponto P_i, numa grelha [1,100]x[1,100], sendo, então, w_{ij} a distância euclidiana entre os pontos P_i e P_j;
  - ✓ os custos, c_{ij}, são obtidos por $c_{ij} = 44722 - w_{ij} + l$, sendo l aleatoriamente gerado no intervalo $[w_i, w_f]$.

Correram-se os algoritmos, registando-se os tempos de execução e os valores óptimos. No caso do Branch & Bound, o algoritmo foi executado com as opções por defeito do Xpress. A relaxação linear foi executada antes do Branch & Bound.

Os testes computacionais foram efectuados num Intel(R)Pentium(R)Dual, CPU T2310, 1,46GHz com 1,0 GB de RAM.

Na realização dos resultados computacionais foi considerado um parâmetro α para gerar as ligações entre os nós. O parâmetro α é utilizado na geração da matriz de adjacência, sendo que, para cada *i* e para cada *j*, caso o valor gerado aleatoriamente pelo processador seja inferior a α, não é gerado o arco *(i, j)*, sendo gerado no caso contrário. De modo a evitar a geração de instâncias triviais, isto é, instâncias das quais resultam problemas cujo caminho mais curto coincide com o caminho de menor peso, foi considerado um parâmetro p para o cálculo do termo independente da restrição de capacidade, w. Mais concretamente: $w = p \times W$, onde W é o peso do caminho mais curto sem restrição de capacidade.

7.1 Resultados Computacionais

De seguida apresentam-se os resultados obtidos para os diversos tipos e grupos anteriormente descritos. Os quadros apresentados encontram-se organizados por tipo e grupo considerados. Para cada um deles são apresentados, inicialmente, os parâmetros utilizados na geração da instância.

Em cada quadro é, ainda, apresentada a informação do número de nós da árvore de Branch & Bound (informação a ler na última coluna).

No final de cada quadro são apresentados quatro gráficos que resumem a informação relativa à relação entre α e:

- os tempos da Relaxação Linear (RL) e da Programação Linear Inteira (PLI);

- d da RL, sendo que d representa a percentagem de desvio do valor obtido pela RL relativamente ao valor óptimo obtido pela programação linear inteira, z^*, isto é: $d = \frac{z^* - z^{LB}}{z^*} \times 100$;

- o número de nós da árvore de Branch & Bound;

- d do Algoritmo B, sendo que d representa a percentagem de desvio do valor obtido pelo Algoritmo B relativamente ao valor óptimo obtido pela programação linear inteira, z^*, isto é: $d = \frac{z^* - z^{LB}}{z^*} \times 100$.

Parâmetros	***w***	***n***	***R***	***p***
	1000	500	20	0,90

α	RL			PLI		PD		Algoritmo B			Nº de nós
	Opt	d	t (seg)	Opt	t (seg)	Opt	t (seg)	UB	d	t (seg)	
0	10,0	0,0	2,0	10	1,9	10	27,5	10	0,0	2,9	1
	11,0	0,0	2,0	11	0,1	11	28,4	13	18,2	2,7	1
	11,0	0,0	1,2	11	2,6	11	28,4	11	0,0	2,7	1
	12,0	0,0	2,2	12	3,7	12	26,4	12	0,0	8,4	1
	11,0	0,0	1,3	11	0,9	11	28,2	11	0,0	2,2	1
0,2	11,0	0,0	0,9	11	2,3	11	27,9	11	0,0	1,6	1
	12,0	0,0	1,4	12	1,5	12	25,2	12	0,0	1,8	1
	12,0	0,0	1,0	12	1,1	12	27,0	13	8,3	2,6	1
	11,0	0,0	1,0	11	1,6	11	25,2	12	9,1	1,8	1
	13,0	0,0	1,6	13	1,4	13	26,6	13	0,0	2,2	1
0,4	13,0	0,0	0,6	13	0,4	13	25,8	14	7,7	1,5	1
	17,0	0,0	0,6	17	2,1	17	24,4	17	0,0	1,8	1
	15,0	0,0	1,1	15	1,9	15	25,9	15	0,0	1,6	1
	14,0	0,0	0,9	14	2,6	14	24,5	14	0,0	1,0	1
	12,0	0,0	0,9	12	1,1	12	28,1	13	8,3	1,9	1
0,6	20,0	0,0	0,4	20	0,7	20	26,0	23	15,0	2,0	1
	20,0	0,0	0,5	20	0,8	20	23,5	21	5,0	1,6	1
	18,0	0,0	0,5	18	0,6	18	23,6	19	5,6	1,0	1
	20,0	0,0	0,4	20	0,9	20	23,5	22	10,0	1,3	1
	21,0	0,0	0,6	21	0,5	21	23,6	21	0,0	1,5	1
0,8	37,0	0,0	0,2	37	0,5	37	21,5	37	0,0	0,8	1
	33,0	0,0	0,2	33	0,3	33	21,5	34	3,0	0,5	1
	38,0	0,0	0,2	38	0,8	38	21,3	38	0,0	1,1	1
	39,0	0,0	0,2	39	0,4	39	21,3	41	5,1	0,5	1
	36,0	0,0	0,2	36	0,5	36	21,4	37	2,8	0,5	1

Quadro I – Resultados computacionais para Tipo I – GI em pequenas dimensões

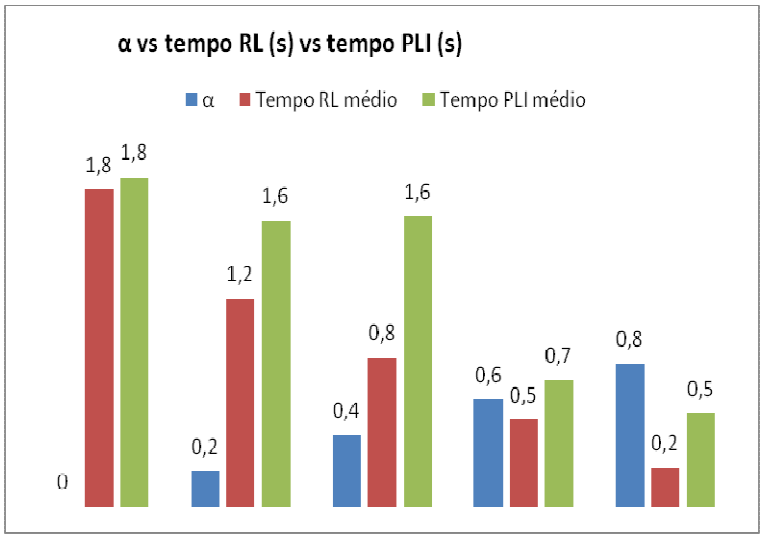


Gráfico 1

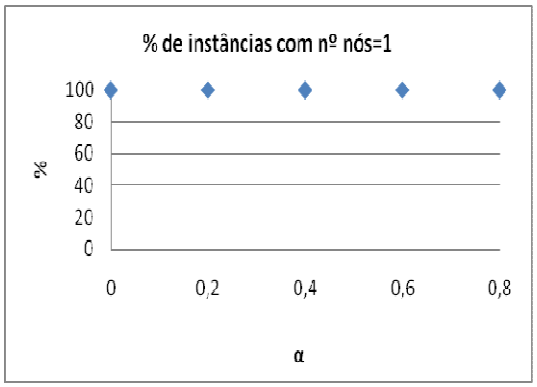


Gráfico 2

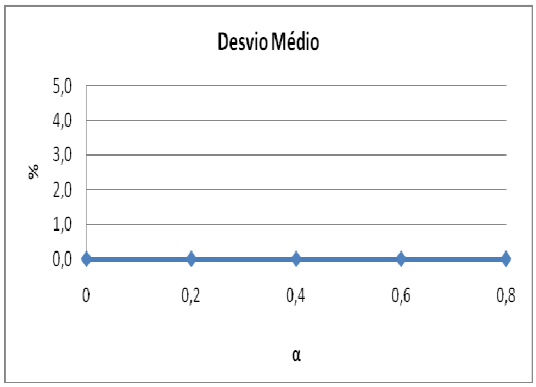


Gráfico 3

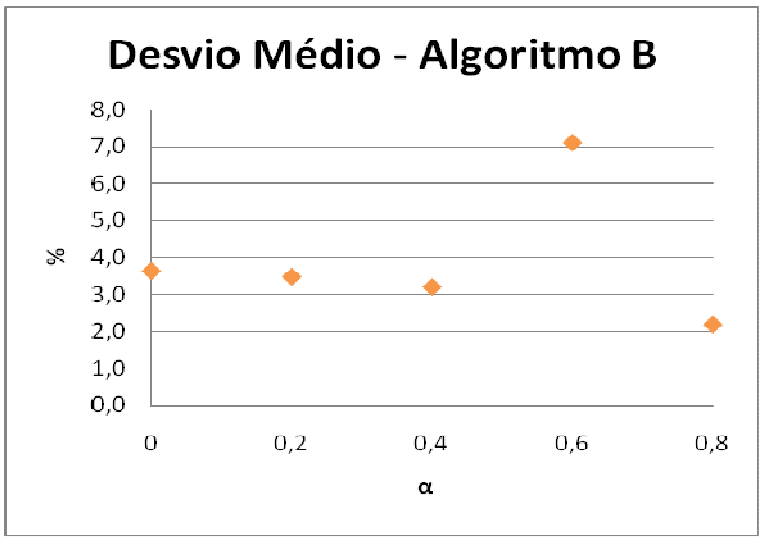


Gráfico 4

Parâmetros	***w***	***n***	***R***	***p***
	3000	1000	500	0,90

α	RL			PLI		Algoritmo B			Nº de nós
	Opt	d	t (seg)	Opt	t (seg)	UB	d	t (seg)	
0	10396,4	0,0	11,9	10401	30,0	84981	717,0	45,5	19
	10415,6	0,1	11,0	10421	7,8	84694	712,7	46,4	45
	10345,3	0,6	11,7	10407	59,5	83456	701,9	46,8	58
	10343,1	0,2	11,2	10360	890,8	84986	720,3	45,7	2567
	10378,7	0,1	11,6	10386	388,4	85094	719,3	45,3	1574
0,2	10285,3	0,1	9,4	10294	286,6	84540	721,3	45,9	1339
	10466,6	0,0	10,8	10467	4,6	84892	711,0	46,1	1
	10445,6	0,1	10,7	10455	769,8	84701	710,1	45,6	1692
	10605,4	0,1	7,4	10614	503,9	84931	700,2	46,1	1398
	10706,5	0,0	8,9	10706	575,3	83521	680,1	46,9	1425
0,4	11093,9	0,1	6,2	11103	39,7	84700	662,9	46,6	491
	10989,3	0,1	8,7	11003	41,3	84409	667,1	46,1	277
	11109,3	0,1	6,2	11117	42,3	83877	654,5	46,3	581
	10935,5	0,2	8,5	10956	535,5	84519	671,4	45,8	2498
	10927,7	0,7	8,1	11002	254,7	81987	645,2	46,8	784
0,6	11388,6	0,2	3,9	11414	501,0	82924	626,5	45,3	1493
	11773,0	0,2	4,7	11795	40,0	84055	612,6	45,5	507
	11790,3	0,1	3,7	11804	79,5	84227	613,5	45,8	1217
	11567,4	0,9	3,5	11678	58,6	82176	603,7	45,7	527
	11401,2	0,1	3,6	11410	498,8	81895	617,7	45,2	1457
0,8	13835,3	0,2	1,6	13864	6,8	80587	481,3	45,5	67
	12061,1	0,3	1,8	12097	93,9	80361	564,3	45,9	1646
	12991,0	0,2	2,7	13021	72,5	81761	527,9	45,6	712
	13976,2	0,0	2,3	13980	7,5	80214	473,8	46,5	78
	13994,8	0,2	1,9	14016	12,4	79728	468,8	46,4	54

Quadro II – Resultados computacionais para Tipo I – GI para grandes dimensões

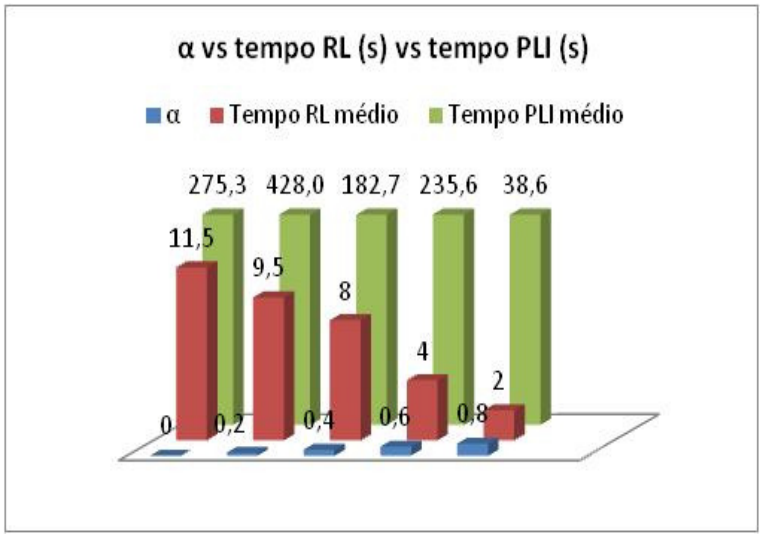


Gráfico 5

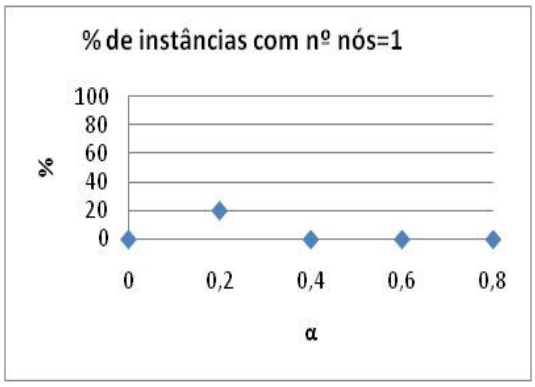


Gráfico 6

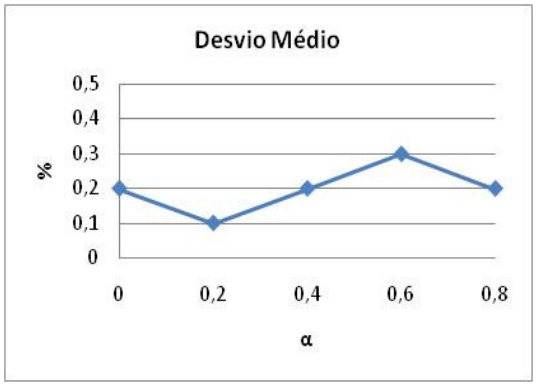


Gráfico 7

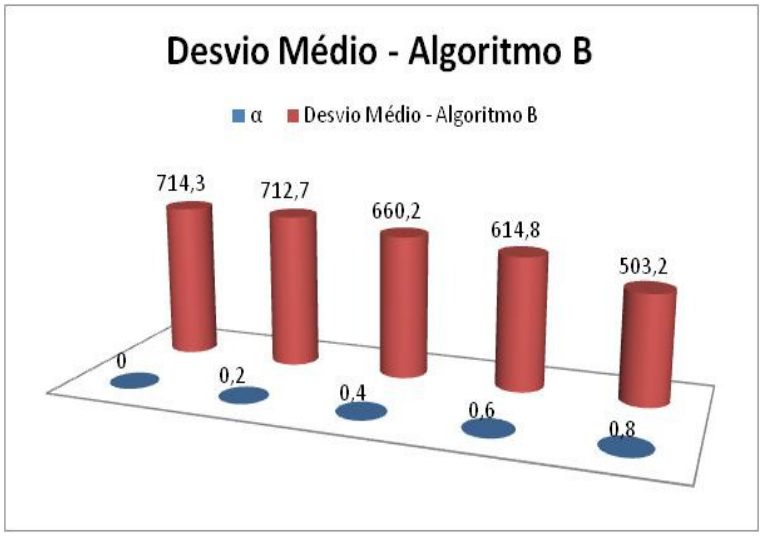


Gráfico 8

Parâmetros	***w***	***n***	***R***	***p***
	1000	500	20	0,90

α	RL			PLI		PD		Algoritmo B			Nº de nós
	Opt	d	t (seg)	Opt	t (seg)	Opt	t (seg)	UB	d	t (seg)	
0	2,1	30,9	0,7	3	1,6	3	15,3	9	200,0	2,5	1
	3,0	0,0	0,4	3	0,1	3	18,0	6	100,0	1,9	1
	1,0	0,0	0,4	1	0,078	1	17,1	2	100,0	0,5	1
	2,0	33,3	0,7	3	1,6	3	15,8	5	66,7	0,4	1
	3,0	0,0	0,4	3	0,1	3	15,9	5	66,7	3,1	1
0,2	5,0	0,0	0,3	5	0,2	5	15,2	8	60,0	0,5	1
	4,0	0,0	0,2	4	0,2	4	15,2	6	50,0	0,7	1
	2,0	0,0	0,3	2	0,3	2	16,0	4	100,0	0,5	1
	3,0	0,0	0,4	3	0,2	3	16,3	6	100,0	1,4	1
	3,0	0,0	0,4	3	0,2	3	16,0	7	133,3	1,6	1
0,4	7,0	0,0	0,2	7	0,0	7	14,5	12	71,4	1,3	1
	8,0	0,0	0,3	8	0,2	8	14,7	12	50,0	1,4	1
	7,2	10,4	0,5	8	0,5	8	14,9	13	62,5	1,6	1
	7,0	0,0	0,2	7	0,4	7	15,3	11	57,1	1,7	1
	8,0	0,0	0,3	8	0,0	8	16,1	11	37,5	0,8	1
0,6	10,4	5,5	0,3	11	0,1	11	13,9	18	63,6	2,3	1
	13,0	0,0	0,3	13	0,1	13	14,2	16	23,1	2,1	1
	15,0	0,0	0,2	15	0,0	15	15,2	17	13,3	1,0	1
	11,0	0,0	0,2	11	0,1	11	13,5	13	18,2	0,9	1
	15,0	0,0	0,2	15	0,0	15	15,9	18	20,0	0,5	1
0,8	20,8	5,7	0,6	22	0,1	22	13,0	32	45,5	2,5	1
	23,0	4,2	0,2	24	0,2	24	13,7	30	25,0	2,1	1
	24,0	0,2	0,3	24	0,2	24	14,7	31	29,2	2,3	1
	20,0	9,1	0,5	22	0,1	22	14,2	31	40,9	1,6	1
	25,0	0,0	0,1	25	0,031	25	14,83	27	8,0	1,0	1

Quadro III – Resultados computacionais para Tipo I – GII em pequenas dimensões

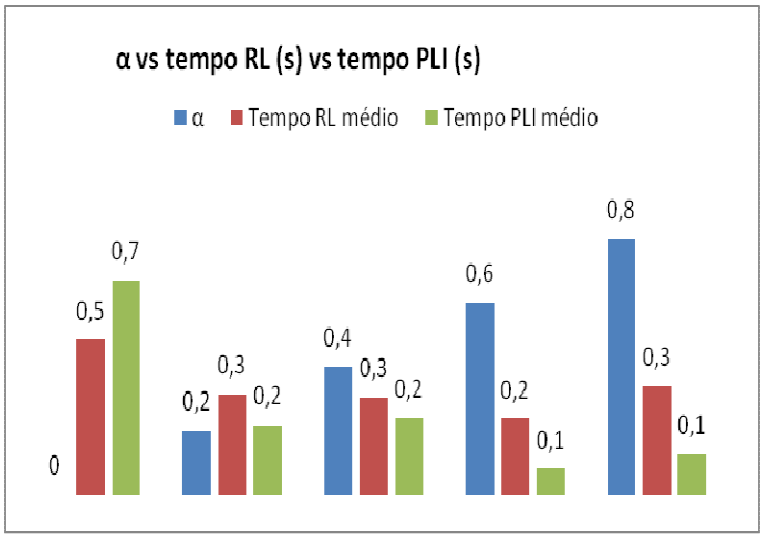


Gráfico 9

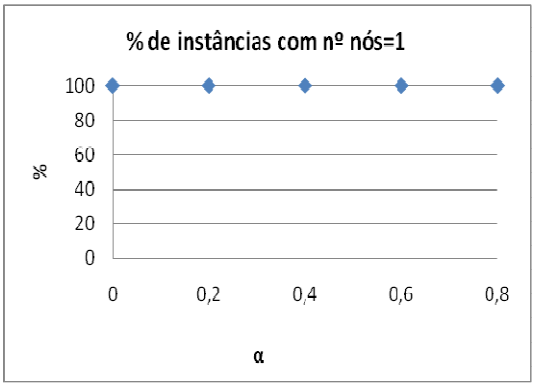


Gráfico 10

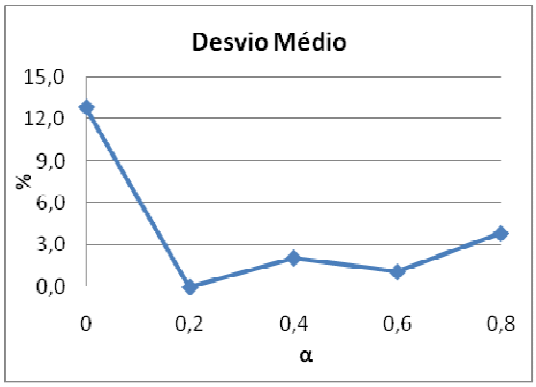


Gráfico 11

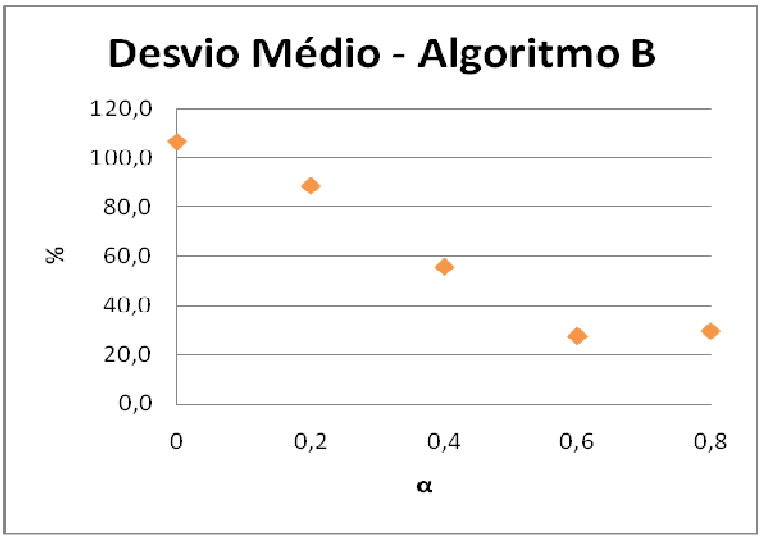


Gráfico 12

Parâmetros	***w***	***n***	***R***	***p***
	3000	1000	500	0,90

α	RL			PD		Algoritmo B			Nº de nós
	LB	d	t (seg)	Opt	t (seg)	UB	d	t (seg)	
0	845,5	0,5	13,3	850	2,5	860	1,2	3,2	1
	846,5	1,7	3,1	861	1,5	876	1,7	2,2	1
	786,5	1,6	2,9	799	2,199	860	7,6	3,2	1
	721,5	1,0	2,4	729	1,7	736	1,0	2,2	1
	815,9	3,7	3,4	847	2,2	1251	47,7	12,3	1
0,2	1006,7	3,8	3,0	1046	3,6	1151	10,0	3,5	1
	911,8	0,5	1,8	916	1,1	930	1,5	2,2	1
	862,3	3,0	2,7	889	3,5	921	3,6	5,3	1
	1002,7	2,1	2,0	1024	1,5	1106	8,0	9,2	1
	1079,5	4,3	2,9	1128	2,7	1300	15,2	6,4	1
0,4	1055,7	0,3	1,4	1059	0,8	1135	7,2	3,1	1
	1288,1	2,2	1,5	1317	2,1	1458	10,7	11,3	1
	1263,0	0,0	1,2	1263	0,4	1263	0,0	2,2	1
	975,8	7,8	2,3	1058	2,0	1200	13,4	2,2	1
	1136,1	3,3	1,9	1175	3,2	1525	29,8	12,5	1
0,6	1517,1	0,4	1,0	1523	1,0	1528	0,3	2,2	1
	1583,2	1,7	1,2	1611	1,2	1653	2,6	4,3	1
	1472,5	3,0	1,3	1518	0,8	1652	8,8	5,3	1
	1982,4	0,9	0,9	2001	0,5	2006	0,2	2,2	1
	1362,2	1,3	0,8	1380	1,2	1515	9,8	2,1	1
0,8	2590,8	5,1	0,6	2729	0,4	2729	0,0	12,4	1
	2285,4	1,3	0,7	2316	0,5	2316	0,0	6,1	1
	2905,7	0,8	0,7	2930	0,4	3320	13,3	7,2	1
	2407,3	1,4	0,6	2442	0,3	2557	4,7	5,1	1
	2707,0	0,1	0,6	2711	0,4	2711	0,0	2,2	1

Quadro IV – Resultados computacionais para Tipo I – GII para grandes dimensões

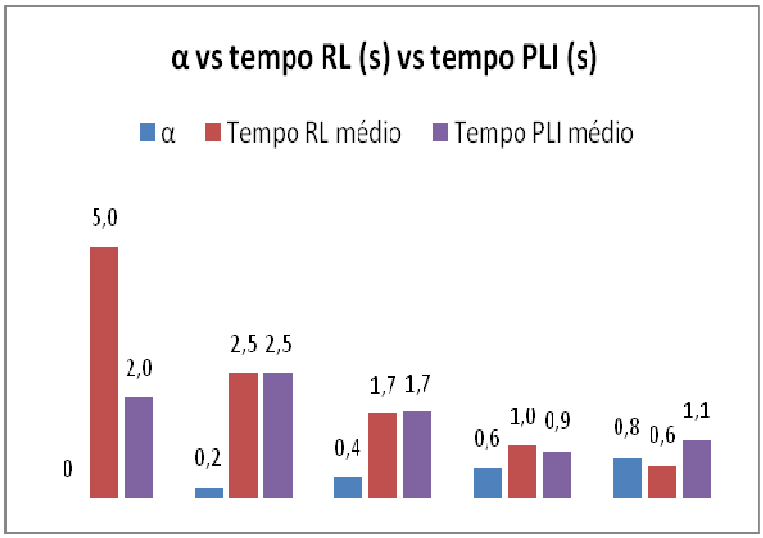


Gráfico 13

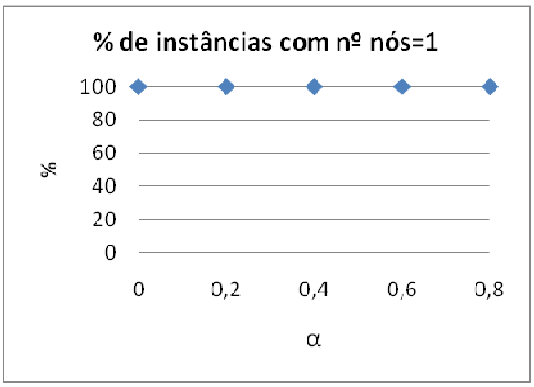


Gráfico 14

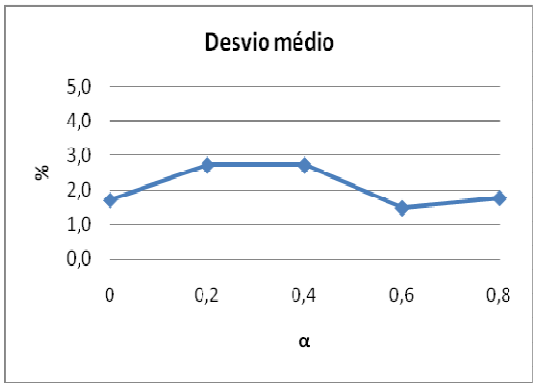


Gráfico 15

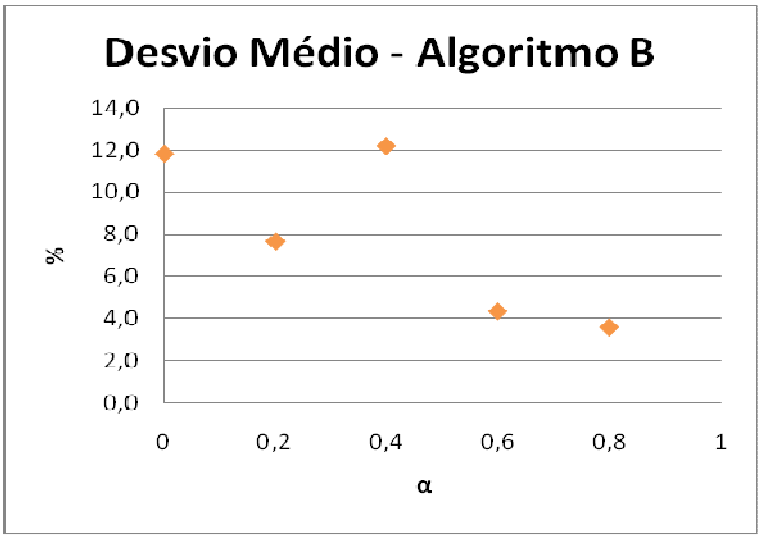


Gráfico 16

Parâmetros	***w***	***n***	***R***	***p***
	1000	500	20	0,90

α	RL			PLI		PD		Algoritmo B			Nº de nós
	Opt	d	t (seg)	Opt	t (seg)	Opt	t (seg)	UB	d	t (seg)	
0	136,0	0,0	0,4	136	0,2	136	19,8	138	1,5	0,6	1
	137,0	0,0	0,3	137	0,1	137	21,0	138	0,7	0,8	1
	135,0	0,0	0,6	135	0,1	135	20,0	137	1,5	0,8	1
	136,0	0,0	0,3	136	1,6	136	17,3	138	1,5	0,6	1
	137,0	0,0	0,5	137	0,5	137	18,5	138	0,7	0,6	1
0,2	138,1	0,6	0,5	139	0,2	139	17,6	139	0,0	0,8	1
	136,6	0,3	0,3	137	0,4	137	16,8	138	0,7	3,4	1
	137,0	0,0	0,4	137	0,6	137	16,5	140	2,2	1,3	1
	139,0	0,0	0,5	139	0,3	139	18,0	141	1,4	2,5	1
	136,8	0,9	0,6	138	0,5	138	15,7	142	2,9	3,4	1
0,4	140,6	0,3	0,4	141	0,2	141	15,1	146	3,5	3,4	1
	139,3	0,5	0,4	140	0,2	140	15,5	144	2,9	3,1	1
	139,4	0,4	0,4	140	0,2	140	15,5	140	0,0	3,1	1
	141,2	0,6	0,3	142	0,2	142	17,5	145	2,1	2,5	1
	141,2	0,5	3,4	142	0,2	142	18,2	144	1,4	0,8	1
0,6	143,3	0,5	0,3	144	0,2	144	16,6	149	3,5	0,5	1
	149,7	0,2	0,3	150	0,1	150	15,2	153	2,0	1,2	1
	150,0	0,0	0,3	150	0,1	150	16,6	155	3,3	1,0	1
	144,3	0,5	0,3	145	0,2	145	16,3	147	1,4	4,3	1
	143,7	0,2	0,3	144	0,1	144	15,8	154	6,9	3,9	1
0,8	165,5	2,6	0,2	170	0,1	170	12,4	174	2,4	6,2	1
	172,0	0,0	0,1	172	0,0	172	13,2	175	1,7	0,5	1
	171,0	0,0	0,1	171	0,0	171	15,1	171	0,0	0,5	1
	161,0	0,0	0,2	161	0,1	161	14,7	164	1,9	0,7	1
	159,9	0,0	0,2	160	0,1	160	15,4	164	2,5	0,8	1

Quadro V – Resultados computacionais para Tipo I – GIII em pequenas dimensões

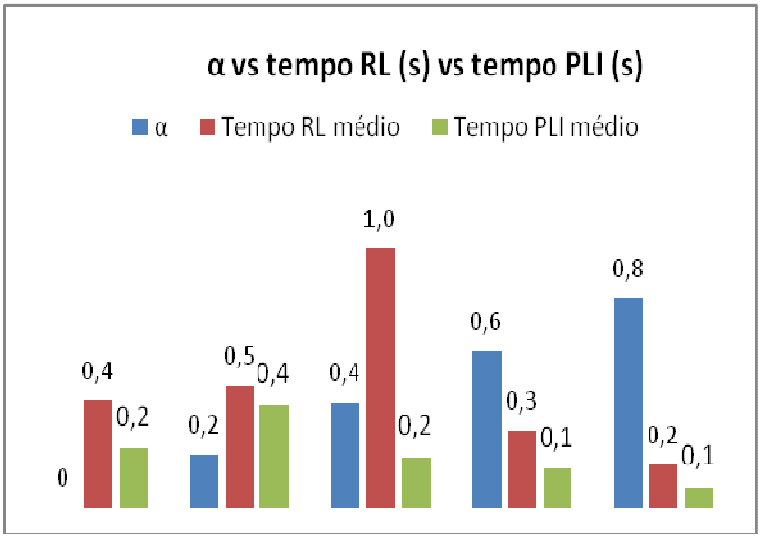


Gráfico 17

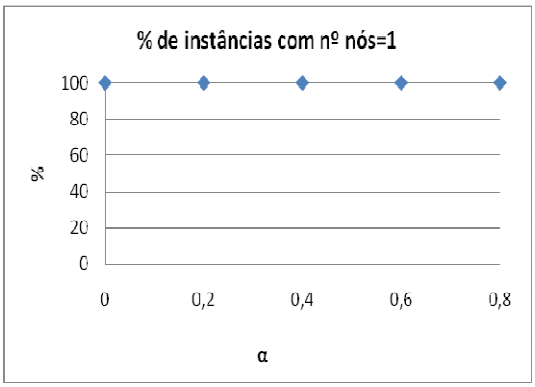


Gráfico 18

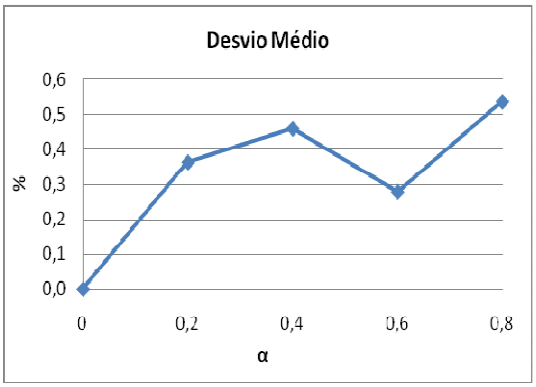


Gráfico 19

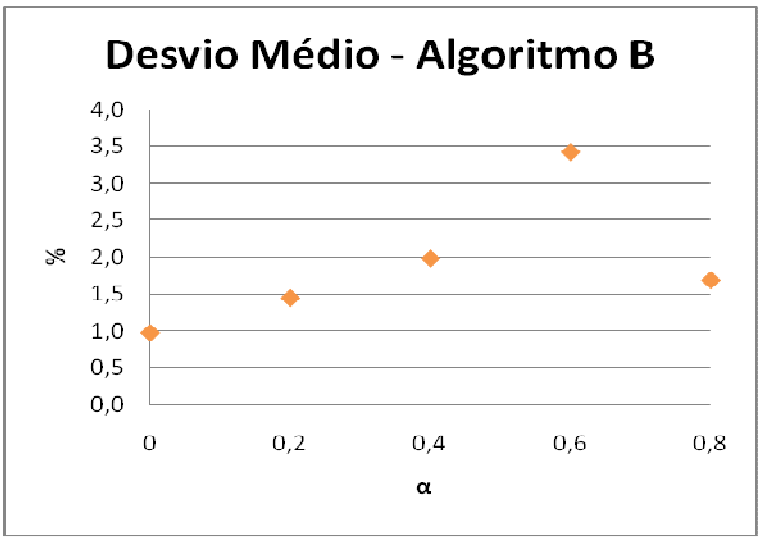


Gráfico 20

Parâmetros	***w***	***n***	***R***	***p***
	3000	1000	5000	0,90

α	RL			PLI		Algoritmo B			Nº de nós
	Opt	d	t (seg)	Opt	t (seg)	UB	d	t (seg)	
0	10208,5	0,0	2,5	10212	1,1	10234	0,2	3,2	1
	10521,0	0,4	2,5	10564	2,6	10576	0,1	3,2	1
	10282,4	0,0	2,2	10284	1,388	10284	0,0	2,2	1
	10511,8	0,3	2,2	10541	3,2	10541	0,0	3,2	1
	10481,9	0,4	3,1	10525	1,5	11224	6,6	12,4	1
0,2	11168,9	0,4	2,0	11215	1,5	11458	2,2	7,3	1
	10784,6	0,2	1,8	10807	2,9	11664	7,9	12,4	1
	10505,6	0,7	2,5	10576	1,4	11077	4,7	10,3	1
	10813,5	0,6	2,4	10884	2,2	10979	0,9	4,3	1
	10677,8	1,1	2,1	10793	4,8	10863	0,6	2,2	1
0,4	11051,8	0,1	1,3	11062	0,8	11062	0,0	2,2	1
	11420,0	1,7	2,2	11613	2,5	12284	5,8	12,5	1
	10825,3	0,6	1,4	10889	1,1	11191	2,8	2,2	1
	10883,6	0,0	1,3	10884	0,2	10884	0,0	2,2	1
	10877,4	0,1	1,4	10885	0,5	11055	1,6	3,2	1
0,6	11342,1	0,7	1,0	11418	1,0	11434	0,1	3,2	1
	12541,4	0,6	1,0	12622	0,6	12671	0,4	2,2	1
	11940,0	0,6	1,1	12017	0,3	12793	6,5	12,4	1
	11593,7	0,6	1,1	11669	0,6	12983	11,3	12,5	1
	12099,6	0,0	0,9	12102	0,4	12102	0,0	2,2	1
0,8	14675,6	1,4	0,6	14884	0,3	15611	4,9	4,2	1
	15789,7	1,1	0,6	15960	0,8	16571	3,8	0,4	1
	16675,5	0,2	0,6	16711	0,2	17130	2,5	2,2	1
	14706,7	0,4	0,6	14760	0,2	14760	0,0	4,2	1
	14566,8	0,1	0,6	14576	0,2	14589	0,1	4,3	1

Quadro VI – Resultados computacionais para Tipo I – GIII para grandes dimensões

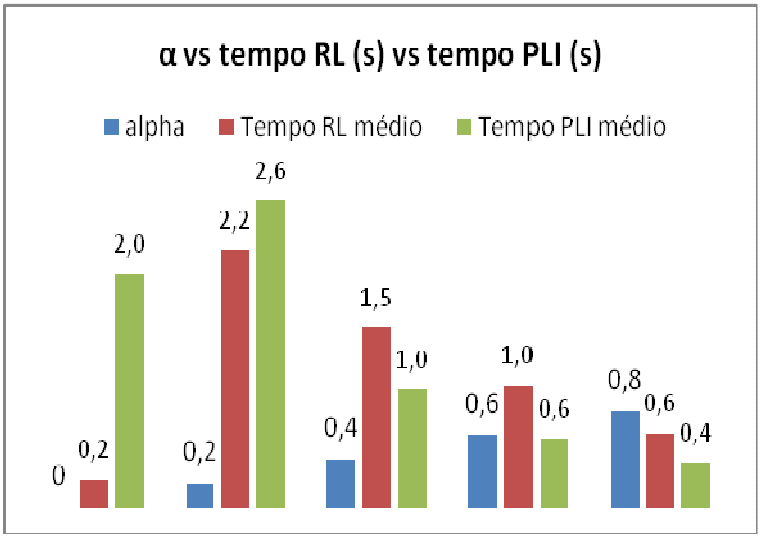


Gráfico 21

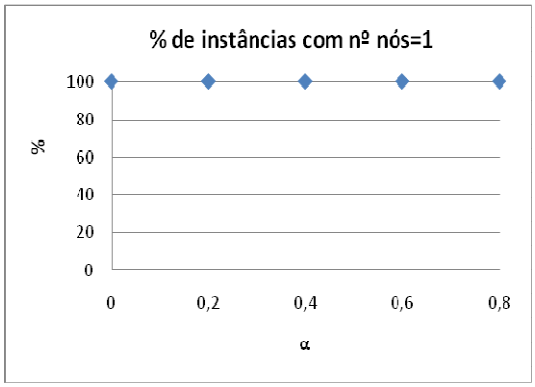


Gráfico 22

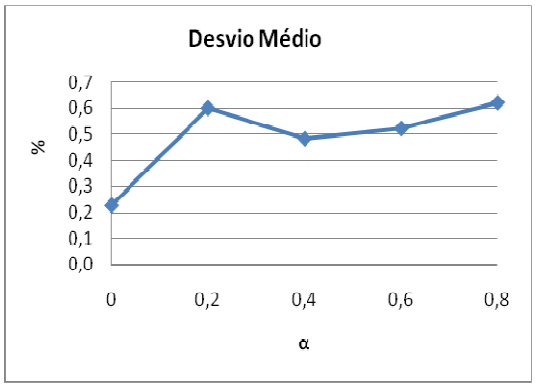


Gráfico 23

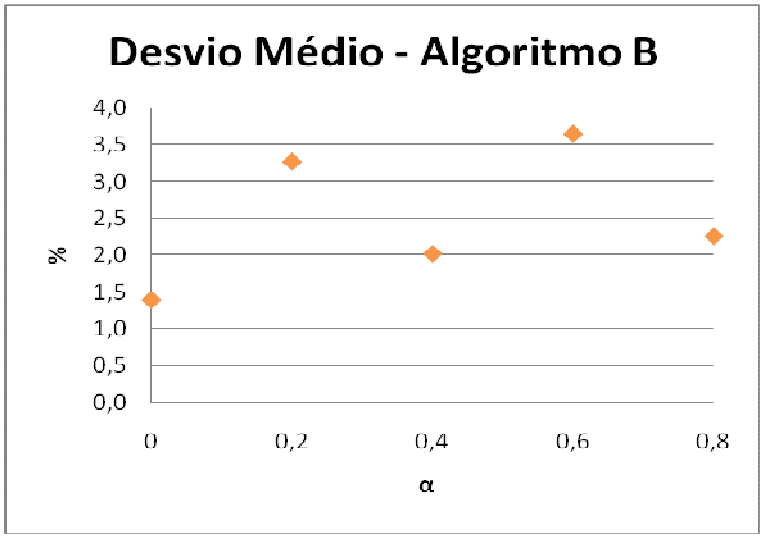


Gráfico 24

Parâmetros	***w***	***n***	***R***	***p***
	1000	500	20	0,90

α	RL			PLI		Algoritmo B			Nº de nós
	Opt	d	t (seg)	Opt	t (seg)	UB	d	t (seg)	
0	204,0	0,0	51,1	204	8,4	6491	3081,9	11,6	1
	6,0	0,0	5,7	6	0,5	6	0,0	1,8	1
	6,0	0,0	2,1	6	5,2	6	0,0	0,9	1
	1002,0	0,0	54,1	1002	7,0	6493	548,0	11,5	1
	1002,0	0,0	56,3	1002	6,3	6491	547,8	12,9	1
0,2	503,0	0,0	39,5	503	8,3	6491	1190,5	11,5	1
	6,0	0,0	4,0	6	0,3	6	0,0	0,6	1
	503,0	0,4	39,8	505	59,5	6491	1185,3	11,6	1
	503,0	0,0	35,1	503	11,5	6493	1190,9	11,5	1
	6,0	0,0	4,0	6	3,7	6	0,0	1,1	1
0,4	503,0	0,0	28,6	503	81,4	6493	1190,9	11,8	1
	389,0	0,5	27,0	391	67,3	649,1	66,0	11,6	1
	6,0	0,0	3,1	6	2,7	6	0,0	1,1	1
	503,0	0,0	35,0	503	4,2	6493	1190,9	11,5	1
	6,0	0,0	3,0	6	2,2	6	0,0	0,6	1
0,6	503,0	0,4	19,1	505	33,5	6495	1186,1	11,7	1
	8,0	0,0	2,3	8	1,1	8	0,0	17,8	1
	503,0	0,0	11,3	503	0,2	6495	1191,3	11,8	1
	503,0	0,0	18,2	503	6,0	6493	1190,9	11,7	1
	8,0	0,0	2,2	8	0,2	8	0,0	0,6	1
0,8	14,4	9,8	1,6	16	1,2	16	0,0	1,1	1
	10,4	13,7	1,5	12	1,2	14	16,7	0,8	1
	9,1	24,4	1,4	12	1,3	14	16,7	1,9	1
	27,4	8,6	4,5	30	1,2	33	10,0	3,0	1
	12,6	9,8	1,5	14	1,3	18	28,6	2,1	1

Quadro VII – Resultados computacionais para Tipo II – GI para pequenas dimensões

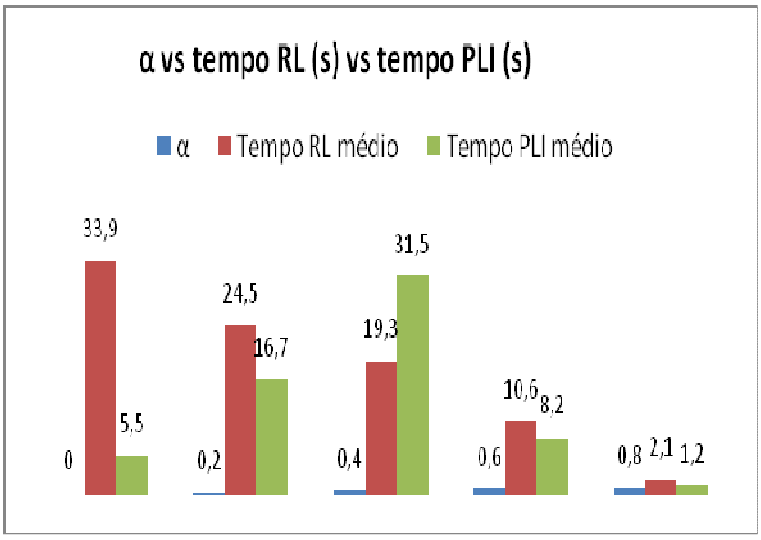


Gráfico 25

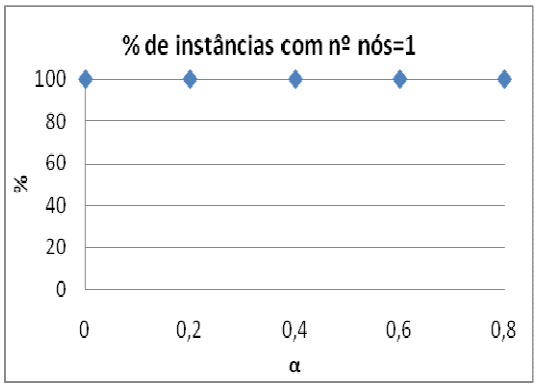


Gráfico 26

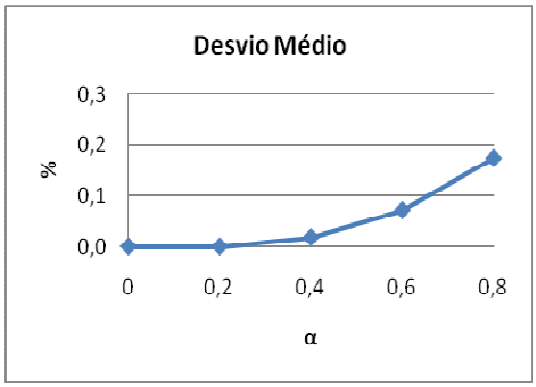


Gráfico 27

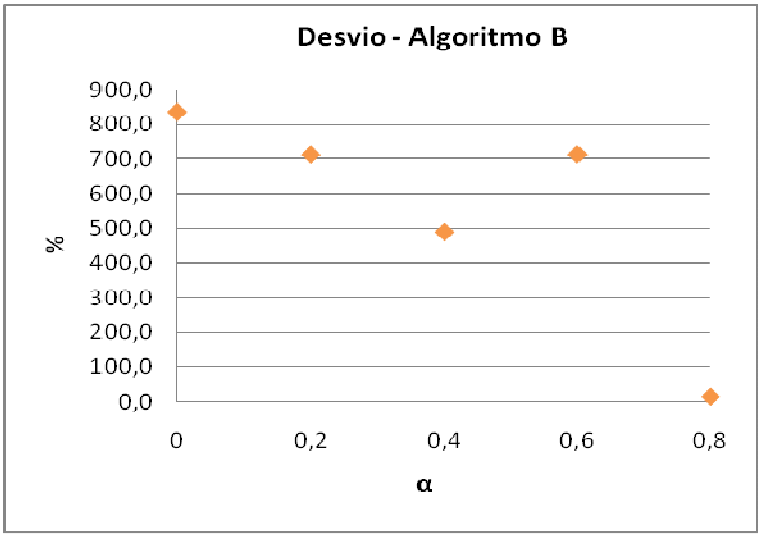


Gráfico 28

Parâmetros	***w***	***n***	***R***	***p***
	3000	1000	5000	0,90

α	RL			PD		Algoritmo B			Nº de nós
	Opt	d	t (seg)	Opt	t (seg)	UB	d	t(seg)	
0	42906,0	0,0	102,4	42906	157,1	1662820	3775,5	21,3	58
	13555,3	7,7	78,8	14688	235,7	1662931	11221,	21,2	60
	57718,0	1,1	70,0	58358	155,2	2239925	3738,2	13,7	52
	57256,0	1,1	52,7	57872	204,5	2243489	3776,6	11,5	58
	218158,0	0,3	47,3	218741	1706,7	2244806	926,2	11,6	1000
0,2	255065,0	0,3	44,1	255765	1008,1	2240024	775,8	11,7	538
	131193,0	0,8	42,2	132217	4881,9	2241364	1595,2	11,9	12995
	132500,0	0,4	43,6	133021	852,3	2242449	1585,8	12,5	754
	131300,0	0,7	42,6	132185	3005,7	1942146	1369,3	11,7	997
	224635,0	0,0	44,3	224654	956,3	2234562	894,7	12,3	706
0,4	36992,0	3,4	34,3	38291	96392,0	2243401	5758,8	11,5	75
	9216,9	5,6	3,2	9759	3,0	11117	13,9	0,6	1
	136660,0	0,7	24,9	137662	912,9	2238870	1526,4	11,5	811
	27552,0	0,4	27,6	27443	956,7	34761	26,7	12,6	98
	143772,0	0,5	25,6	144522	735,4	212567	47,1	11,5	432
0,6	13565,6	2,2	3,6	13866	1,9	13866	0,0	0,5	1
	41777,1	3,0	16,5	43090	42,5	2233143	5082,5	11,7	77
	13241,1	17,5	6,8	16046	2,2	18537	15,5	0,5	1
	142341,0	0,8	19,1	143419	344,4	2237597	1460,2	11,4	387
	135160,0	0,6	17,0	135961	74,8	2238983	1546,8	11,5	95
0,8	29001,4	2,5	1,7	29750	1,2	30000	0,8	0,8	1
	34589,1	1,1	1,2	34983	1,1	34983	0,0	0,5	1
	49806,0	1,3	1,3	50455	1,0	50455	0,0	0,6	1
	22501,8	5,0	2,2	23692	1,0	23692	0,0	1,5	1
	32225,3	0,1	2,0	32261	1,0	32261	0,0	0,6	1

Quadro VIII – Resultados computacionais para Tipo II – GI para grandes dimensões

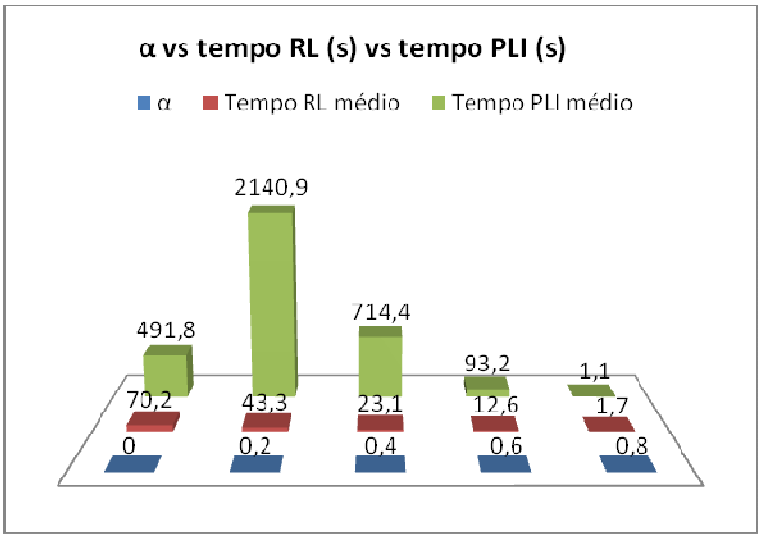


Gráfico 29

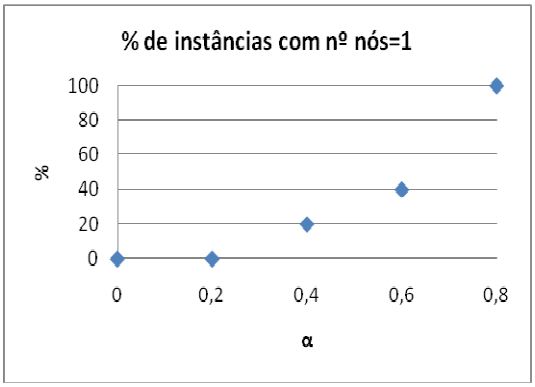


Gráfico 30

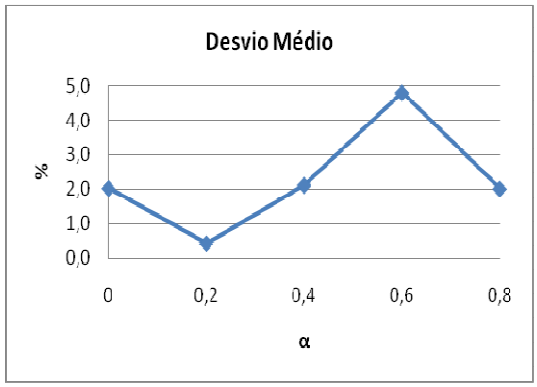


Gráfico 31

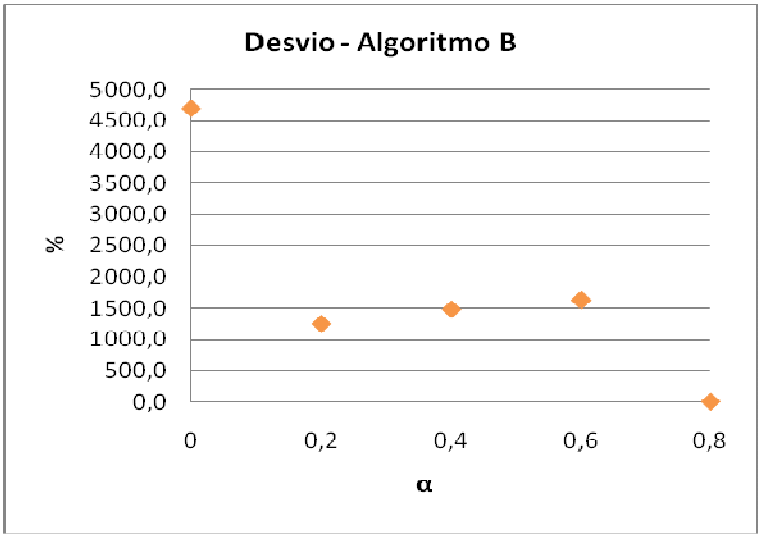


Gráfico 32

Parâmetros	***w***	***n***	***R***	***p***
	1000	500	20	0,95

α	RL			PLI		Algoritmo B			Nº de nós
	Opt	d	t (seg)	Opt	t (seg)	UB	d	t (seg)	
0	3493,0	0,0	10,6	3493	0,5	3493	0,0	0,7	1
	3493,0	0,0	16,4	3493	4,3	3505	0,3	3,2	1
	3493,0	0,0	11,0	3493	1,2	3497	0,1	3,3	1
	3493,0	0,0	11,2	3493	1,3	3504	0,3	3,2	1
	3493,0	0,0	11,0	3493	1,8	3493	0,0	1,3	1
0,2	3493,0	0,0	8,8	3493	1,0	3493	0,0	1,9	1
	3493,0	0,0	8,8	3493	2,7	3493	0,0	1,1	1
	3493,0	0,0	8,8	3493	0,4	3493	0,0	0,6	1
	3493,0	0,0	9,0	3493	0,5	3495	0,1	3,2	1
	3493,0	0,0	8,6	3493	1,6	3493	0,0	0,7	1
0,4	3494,6	0,0	8,6	3496	3,8	3497	0,0	2,1	1
	3495,3	0,0	7,7	3497	2,5	3498	0,0	2,0	1
	3493,0	0,0	6,5	3493	0,7	3493	0,0	2,8	1
	3493,0	0,0	6,5	3493	0,7	3493	0,0	0,8	1
	3493,0	0,0	6,9	3493	0,3	3493	0,0	0,6	1
0,6	3493,9	0,1	4,8	3496	2,0	3496	0,0	0,6	1
	3494,9	0,2	5,4	3501	3,6	3505	0,1	3,1	1
	3493,0	0,0	4,7	3493	0,2	3493	0,0	0,9	1
	3493,0	0,0	4,3	3493	0,2	3493	0,0	0,8	1
	3498,6	0,1	5,9	3503,0	2,4	3521	0,5	3,2	1
0,8	3496,3	0,0	2,3	3497	0,8	3497	0,0	0,6	1
	3497,0	0,0	2,5	3497	0,1	3538	1,2	3,2	1
	3554,9	0,5	3,5	3572	1,4	3594	0,6	3,7	1
	3514,1	0,3	3,0	3523	1,4	3589	1,9	3,2	1
	3495,8	0,1	2,3	3500	1,1	3503	0,1	1,1	1

Quadro IX – Resultados computacionais para Tipo II – GII para pequenas dimensões

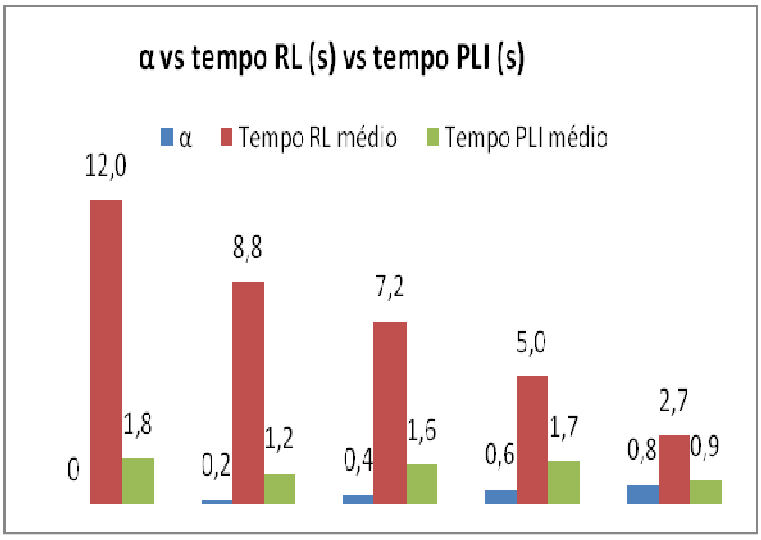


Gráfico 33

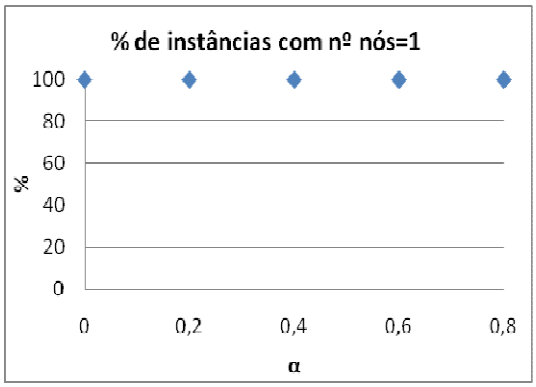


Gráfico 34

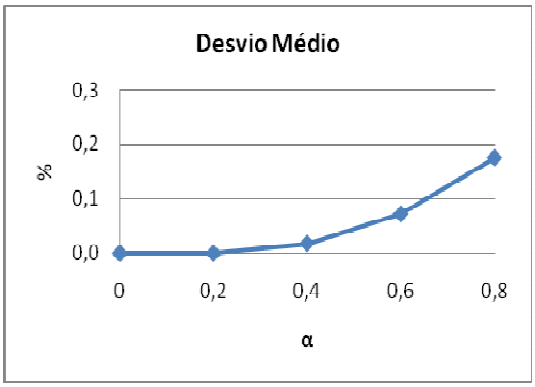


Gráfico 35

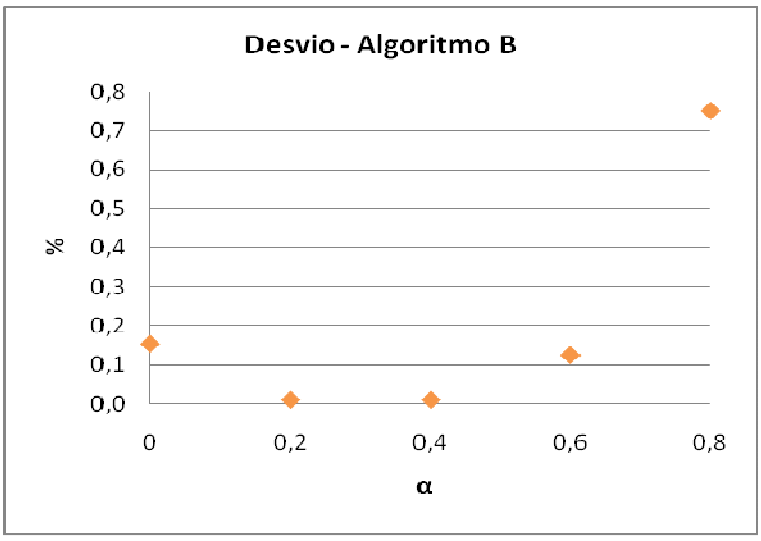


Gráfico 36

Parâmetros	***w***	***n***	***R***	***p***
	3000	1000	5000	0,95

α	RL			PLI		Algoritmo B			Nº de nós
	Opt	d	t (seg)	Opt	t (seg)	UB	d	t (seg)	
0	503346,0	0,4	584,4	505235	95,1	520243	3,0	14,1	1
	504923,0	0,2	752,5	505942	80,6	512230	1,2	14,5	1
	505309,0	0,0	391,9	505410	55,2	505410	0,0	12,3	1
	504367,0	0,3	598,7	505876	76,3	517860	2,4	15,4	1
	504689,0	0,0	311,4	504786	46,2	504786	0,0	6,8	1
0,2	504932,0	0,2	218,2	506112	23,0	506112	0,0	4,6	1
	505417,0	0,3	145,3	506987	21,8	506987	0,0	5,8	1
	507568,0	0,1	154,7	507823	22,9	507823	0,0	5,4	1
	506234,0	0,1	234,7	506754	31,3	506754	0,0	8,4	1
	505043,0	0,1	283,3	505564	33,3	505564	0,0	9,3	1
0,4	508307,0	1,4	96,6	515609	85,8	530058	2,8	17,3	5
	507282,0	0,5	92,2	509867	34,1	534529	4,8	25,3	1
	508337,0	0,3	93,3	509763	19,6	509763	0,0	6,0	1
	503245,0	0,2	91,2	504237	17,3	506754	0,5	19,9	7
	508244,0	0,2	91,9	509058	16,0	509058	0,0	11,5	1
0,6	516908,0	0,6	51,0	520260	34,6	547662	5,3	15,2	1
	505552,0	0,7	32,1	509112	15,5	530777	4,3	13,1	1
	518242,0	0,9	30,7	523069	3,2	523069	0,0	2,4	1
	515634,0	0,4	35,7	517654	12,5	532454	2,9	6,5	1
	511274,0	0,2	29,1	512164	15,4	512948	0,2	3,6	1
0,8	520492,0	0,3	20,0	522232	5,9	531223	1,7	3,2	1
	536542,0	0,2	19,7	537543	5,7	549876	2,3	9,5	1
	526706,0	1,0	15,7	531921	7,2	550006	3,4	12,4	1
	524376,0	0,0	18,8	524302	6,4	532542	1,6	6,5	1
	523493,0	0,9	16,8	528262	5,8	533489	1,0	7,2	1

Quadro X – Resultados computacionais para Tipo II – GII para grandes dimensões

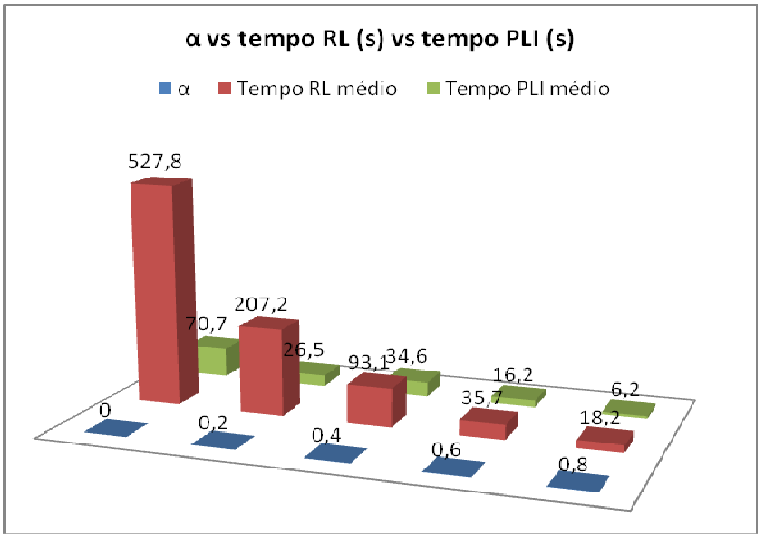


Gráfico 37

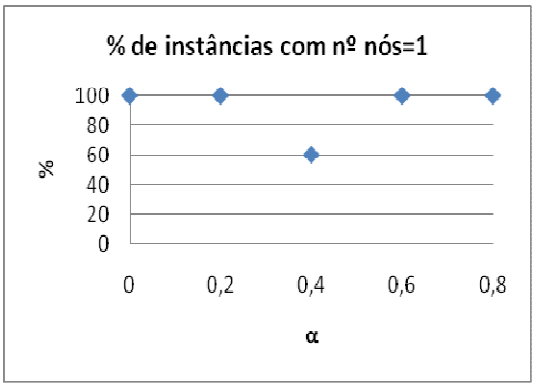


Gráfico 38

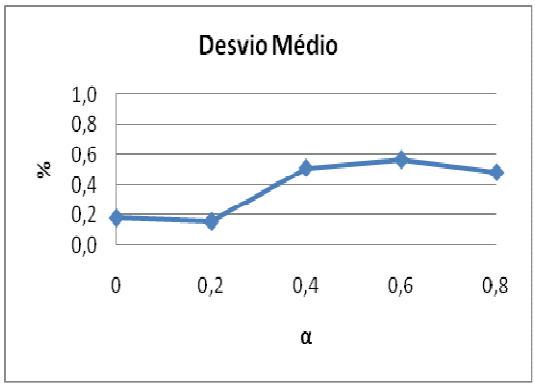


Gráfico 39

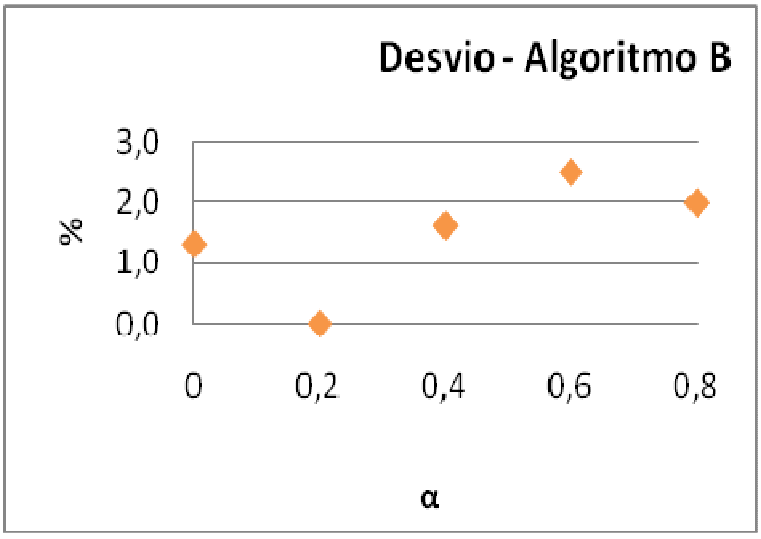


Gráfico 40

Parâmetros	***w***	***n***	***R***	***p***
	1000	500	20	0,90

α	RL			PLI		Algoritmo B			Nº de nós
	Opt	d	t (seg)	Opt	t (seg)	UB	d	t (seg)	
0	26516,7	0,0	10,0	26524	9,8	26875	1,3	11,4	1
	26533,5	0,0	8,4	26537	9,8	26875	1,3	11,5	1
	26501,8	0,1	8,5	26520	12,121	26874	1,3	11,8	1
	26511,2	0,1	8,6	26528	10,0	26878	1,3	11,5	1
	26520,4	0,1	10,1	26538	8,7	26877	1,3	11,8	1
0,2	26519,8	0,1	5,9	26535	5,2	26872	1,3	11,7	1
	26527,8	0,0	6,1	26538	6,4	26875	1,3	11,6	1
	26523,1	0,1	6,6	26541	7,0	26873	1,3	11,7	1
	26561,2	0,0	6,1	26571	4,4	26873	1,1	11,6	1
	26541,0	0,0	6,6	26553	5,6	26874	1,2	11,6	1
0,4	26537,9	0,1	4,1	26558	3,6	26867	1,2	11,6	1
	26498,7	0,1	4,5	26520	5,0	26567	0,2	10,6	1
	26540,8	0,1	5,0	26559	2,7	26874,0	1,2	11,3	1
	26533,2	0,1	4,1	26553	4,1	26872	1,2	11,5	1
	26570,6	0,1	5,6	26592	4,6	26878	1,1	11,6	1
0,6	26577,9	0,1	2,5	26601	2,8	26601	0,0	9,6	1
	26589,3	0,1	2,6	26606	3,2	26873	1,0	11,6	1
	26557,5	0,1	2,8	26571	2,6	26875	1,1	11,5	1
	26640,3	0,1	2,6	26659	4,6	26877	0,8	11,7	1
	26535,0	0,1	3,3	26557	2,1	26878	1,2	11,4	1
0,8	26655,0	0,2	1,4	26696	1,4	26869	0,6	11,4	1
	26541,6	0,1	1,3	26568	0,8	26568	0,0	1,1	1
	26718,8	0,1	1,3	26738	1,0	26781	0,2	10,2	1
	26624,6	0,1	1,5	26660	1,3	26889	0,9	11,0	1
	26567,5	0,1	1,3	26597	1,5	26597	0,0	8,1	1

Quadro XI – Resultados computacionais para Tipo III – GI para pequenas dimensões

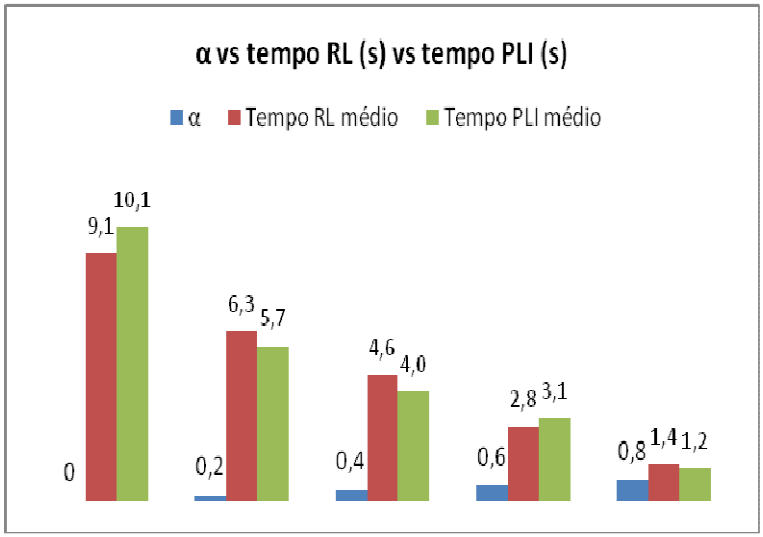


Gráfico 41

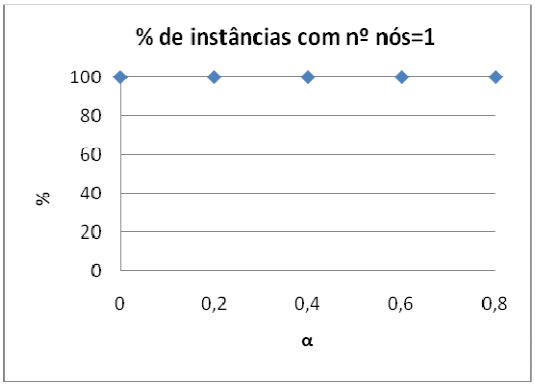


Gráfico 42

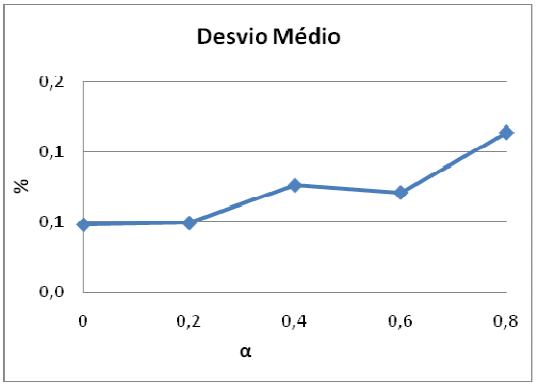


Gráfico 43

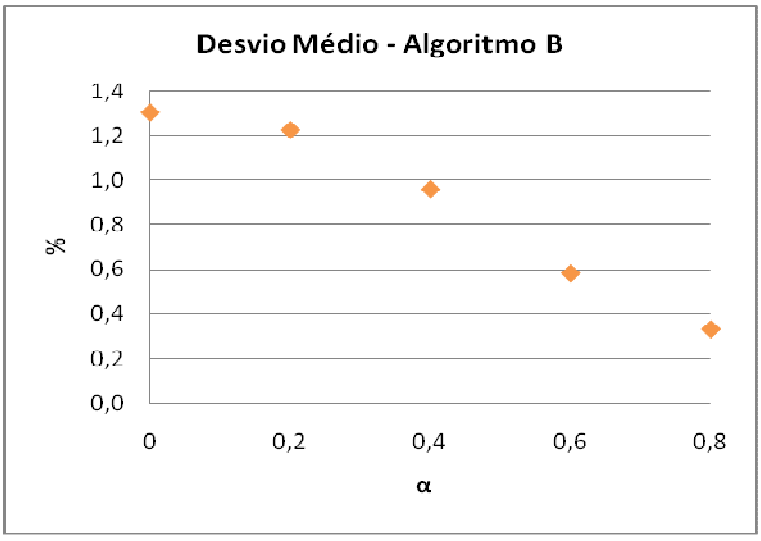


Gráfico 44

Parâmetros	***w***	***n***	***R***	***p***
	3000	1000	5000	0,90

α	RL			PLI		Algoritmo B			Nº de nós
	Opt	d	t (seg)	Opt	t (seg)	UB	d	t (seg)	
0	27154,3	0,1	243,7	27179	51,2	27179	0,0	30,7	1
	26254,0	0,1	176,9	26272	49,5	26272	0,0	12,5	1
	26728,0	0,0	173,2	26730	36,987	26730	0,0	6,7	1
	27321,0	0,1	143,6	27340	32,5	27340	0,0	6,4	1
	27677,0	2,1	139,8	27116	49,6	27116	0,0	7,4	1
0,2	27186,5	1,0	202,2	27471	29,9	27471	0,0	12,9	1
	27135,5	1,2	197,3	27456	39,5	27456	0,0	9,6	1
	27040,5	0,3	170,0	27113	48,8	27113	0,0	5,1	1
	26914,0	0,3	197,3	26983	35,6	26983	0,0	6,8	1
	26874,0	0,0	202,1	26874	32,5	26874	0,0	5,0	1
0,4	27225,8	1,5	50,0	27652	16,8	27652	0,0	41,8	1
	27317,3	1,7	68,5	27798	24,8	27798	0,0	39,7	1
	27128,9	1,0	64,7	27401	21,4	27401	0,0	3,9	1
	26866,7	1,8	45,1	27372	26,3	27372	0,0	10,4	1
	26956,7	0,1	53,7	26997	27,7	26997	0,0	9,4	1
0,6	27544,5	2,8	12,4	28326	11,4	28326	0,0	5,3	1
	27345,5	3,0	9,9	28196	12,4	28196	0,0	4,3	1
	27288,4	0,7	9,3	27473	14,4	27473	0,0	3,2	1
	27215,3	0,7	12,3	27420	12,4	27420	0,0	5,3	1
	27205,3	0,9	10,6	27460	14,2	27460	0,0	4,0	1
0,8	29555,0	3,8	14,0	30728	4,8	30728	0,0	96,5	1
	29678,3	0,3	12,4	29763	4,4	29763	0,0	9,3	1
	28330,6	2,0	4,8	28920	5,4	28920	0,0	2,3	1
	29765,2	2,7	7,2	30578	5,0	30578	0,0	67,4	1
	28916,7	1,7	6,0	29415	5,0	29415	0,0	5,3	1

Quadro XII – Resultados computacionais para Tipo III – GI para grandes dimensões

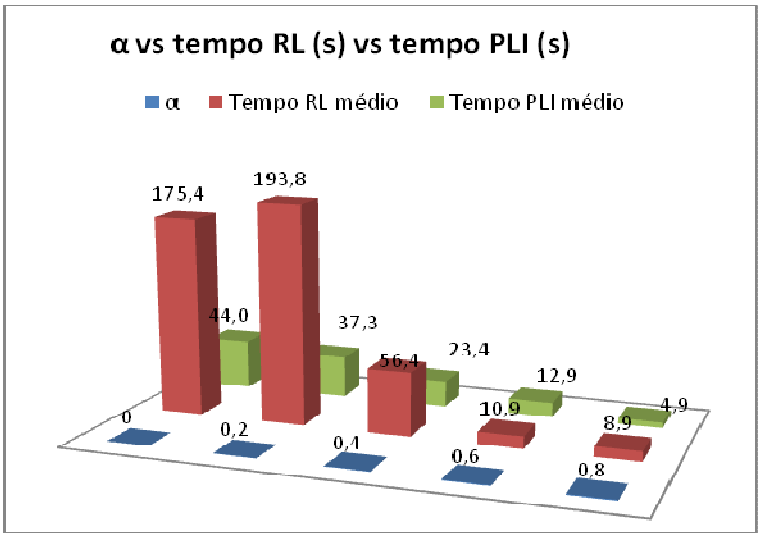


Gráfico 45

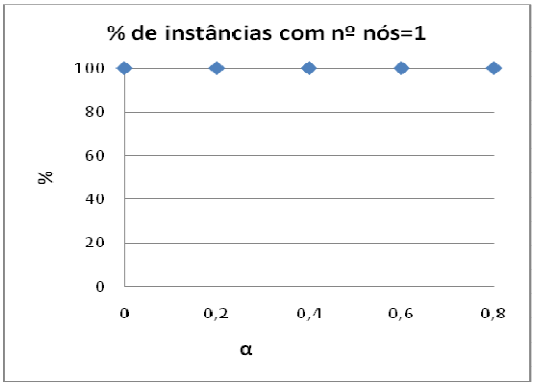


Gráfico 46

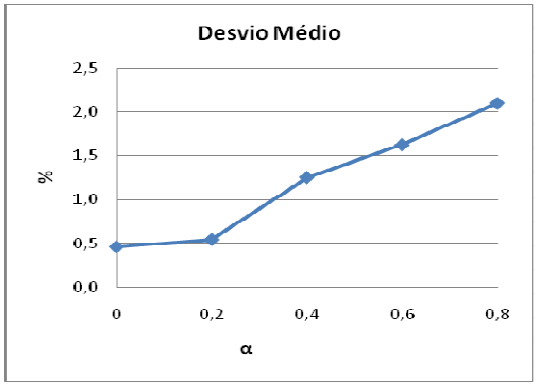


Gráfico 47

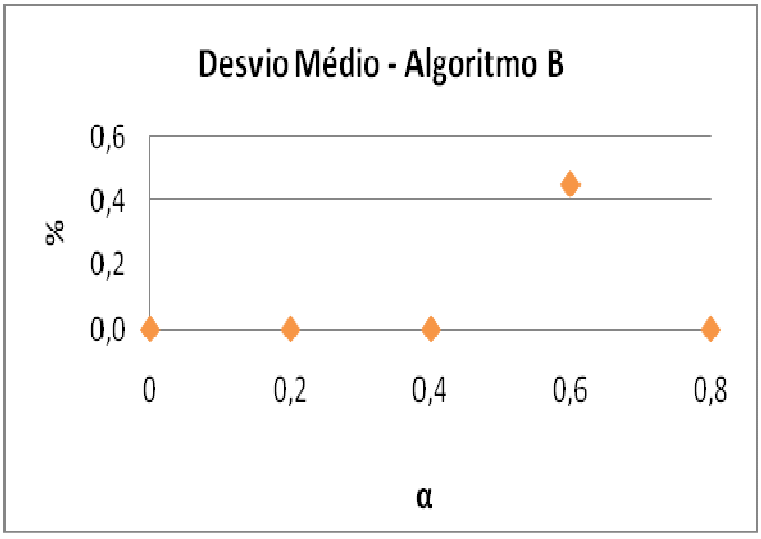


Gráfico 48

7.2 Análise dos resultados

Os testes computacionais foram iniciados tendo como referência o estudo de Pisinger (2005) para o problema do Saco-Mochila. Durante a realização dos testes computacionais verificou-se que para instâncias onde existe correlação positiva e forte entre os custos e os pesos, o caminho mais curto coincide, usualmente, com o caminho de menor peso, sendo este um caso trivial. Os testes preliminares que não foram aqui apresentados permitiram, ainda, concluir que gerando aleatoriamente custos e pesos para pequenas dimensões se obtêm, também, problemas triviais, ocorrendo a situação descrita anteriormente.

7.2.1 Instâncias difíceis

Para caracterizar as instâncias difíceis atendeu-se aos seguintes factores: tempo de execução do Branch & Bound, número de nós da árvore de enumeração do Branch & Bound e o desvio da solução da RL relativamente à solução óptima.

Na globalidade dos testes computacionais realizados, os tempos da PLI são superiores quando se correm instâncias de grandes dimensões do que quando as instâncias são de pequenas dimensões. Esta situação não é bastante evidente para o Tipo I GII e Tipo I GIII.

Quanto ao número de nós usado na árvore Branch & Bound, na grande maioria dos testes realizados foi apenas utilizado um nó. A excepção ocorre, neste caso, em três situações: Tipo I GI (ver Gráfico 6), Tipo II GII (ver Gráfico 30) e Tipo II GII (ver Gráfico 38). Na primeira situação (ver Gráfico 6) quando se utiliza α=0,2 é usado, em 20% dos casos, apenas um nó, sendo que nas restantes situações são usados mais do que um. Na segunda situação (ver Gráfico 30), verifica-se que à medida que o parâmetro α aumenta, a percentagem de testes em que são utilizados mais do que um nó diminui, começando em 100% para α=0, terminando em 0% (o que significa que apenas um nó é utilizado) para α=0,8. Na última situação (ver Gráfico 38), quando α=0,4 é utilizado mais do que um nó em 40% dos casos.

Para α=0 e α=0,8 o desvio médio do valor da RL para dimensões pequenas é substancialmente superior ao desvio médio da RL para dimensões grandes (ver Gráfico 49). Já no que concerne aos restantes valores do parâmetro α, acontece o contrário, isto é, o valor do desvio médio da RL para dimensões pequenas é menor do que para dimensões grandes. Realce-se, aqui, o facto do desvio, em todos os casos, não ser muito grande, ficando, sempre, abaixo dos 3%.

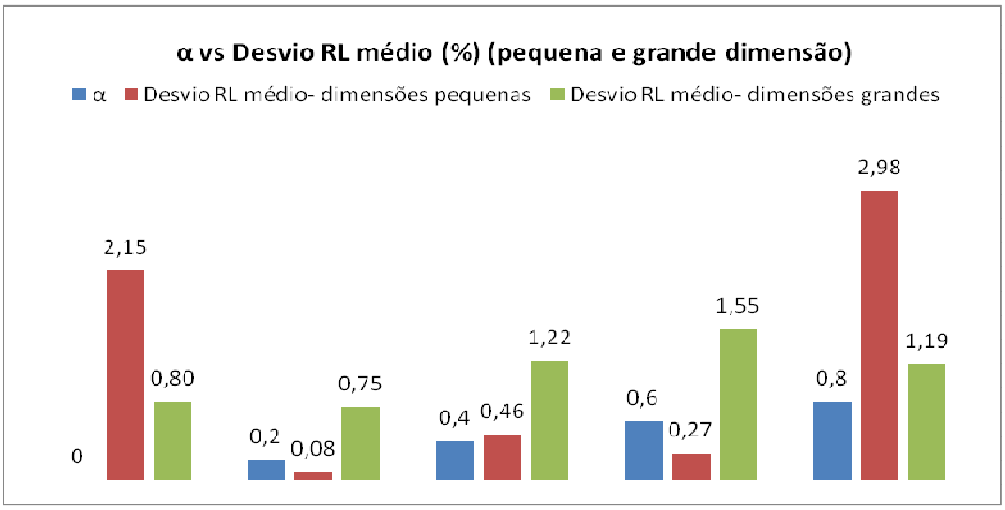


Gráfico 49

Comparando, agora, os tempos médios de execução da PLI para pequenas e grandes dimensões, verifica-se que os mesmos são bastante mais elevados quando as dimensões são grandes. Quando α=0,8 a diferença não é significativa, sendo bastante significativa quando α=0,2 (ver Gráfico 50).

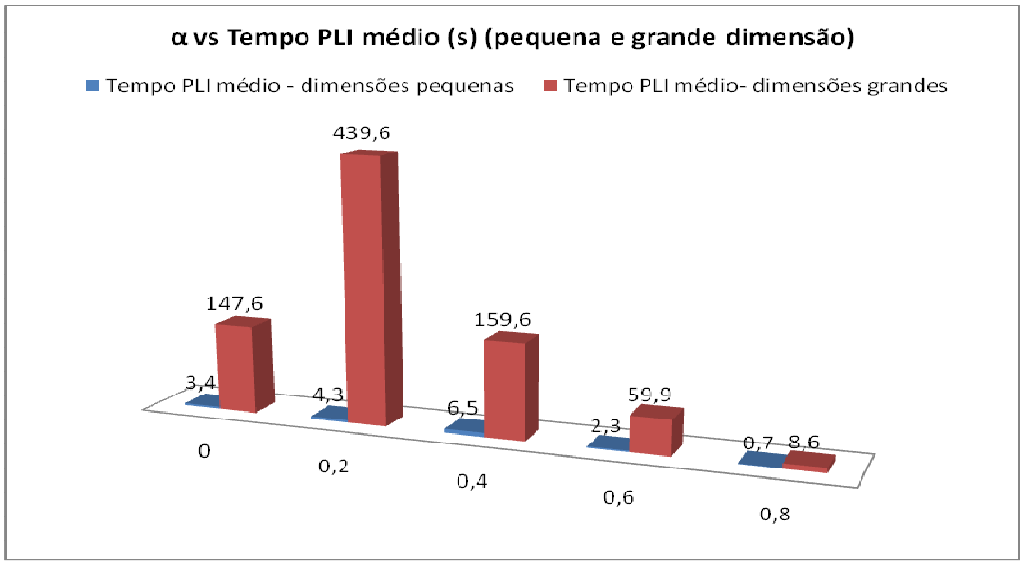


Gráfico 50

Quanto ao número médio de nós usado no Branch & Bound, como se pode observar no Gráfico 52, para dimensões pequenas é sempre 1. Para dimensões grandes verifica-se a utilização de um número médio de nós bastante superior a 1, sendo esta situação mais evidente quando α=0,2.

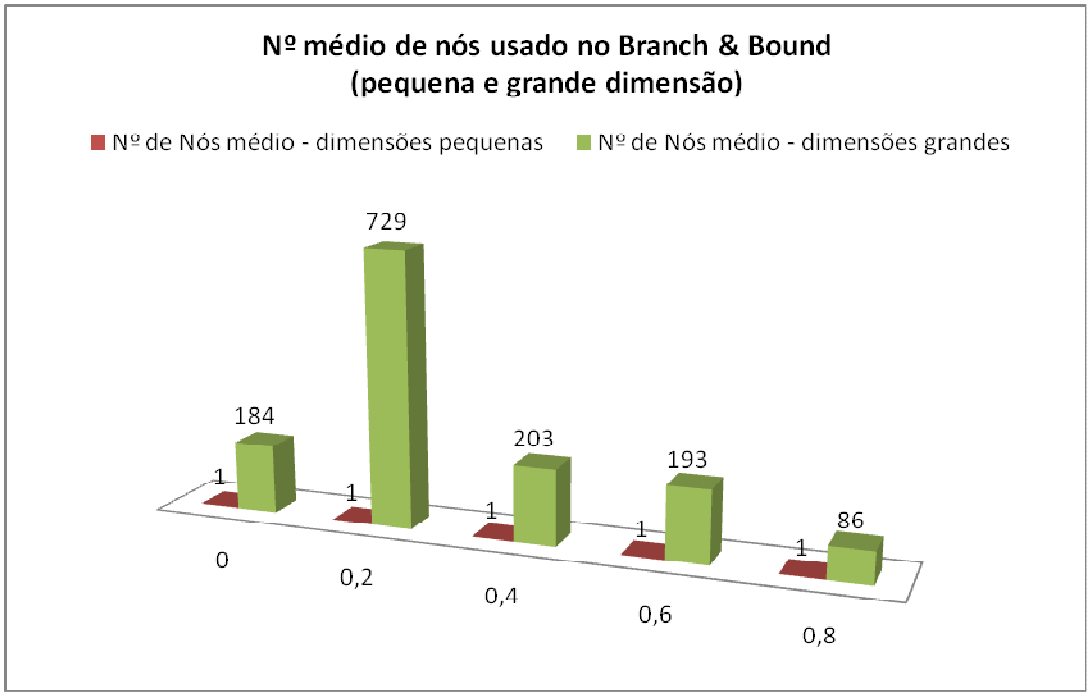


Gráfico 51

Verifica-se, ainda, que as instâncias em que existe uma relação inversa entre os pesos e os custos são mais difíceis do que aquelas em que essa relação é directa. Por exemplo, os

grupos de testes computacionais Tipo I GI e Tipo II GII utilizam, na grande maioria dos testes, mais do que um nó (ver Quadros 2 e 8, bem com Gráficos 6 e 30).

É possível concluir, deste estudo computacional, que as instâncias geradas pelas tipologias Tipo I e Tipo II são mais difíceis do que as instâncias geradas pela tipologia Tipo III. Por outras palavras, pode concluir-se que as instâncias em que o grafo foi estruturado por níveis e as instâncias em que pesos e custos são gerados aleatoriamente num intervalo, tendo em consideração a distância entre os índices dos nós, são mais difíceis do que as instâncias em que os pesos e os custos são gerados tendo em conta a distância euclidiana entre os nós.

A dimensão dos problemas é, também, factor de distinção no que concerne à dificuldade das instâncias. Como poderá ser observado em cada par de tabelas para o mesmo tipo e grupo, quando a dimensão do problema é grande, mais difícil é a instância. Esta informação pode, ainda, ser retirada da leitura dos gráficos 49, 50 e 51. A densidade do grafo é, ainda, relevante no que concerne à dificuldade das instâncias. Instâncias geradas com maior densidade, isto é, para valores do parâmetro α mais pequenas, mostraram-se mais difíceis.

Observou-se, ainda, que a Programação Dinâmica, quando comparada com o Branch & Bound, demora muito mais tempo, sendo menos eficiente, portanto.

7.2.2 Algoritmo B

Quanto ao Algoritmo B, o mesmo apresenta, na globalidade, tempos de execução baixos, ocorrendo a maior excepção quando pesos e custos se relacionam de forma inversa. Isto indica que, nessas circunstâncias, apenas nas últimas iterações (isto é, quando a função objectivo tem em consideração essencialmente os pesos e não os custos) é encontrada uma solução admissível.

Como se pode observar nos Gráficos 28 e 32, é para o Tipo II GI que o valor do Algoritmo B apresenta um desvio maior, atingido valores muito elevados, chegando a atingir os 5000% para dimensões grandes.

Também para o Tipo I GI estes desvios são muito elevados

Com a excepção destas instâncias que revelam características muito específicas, o Algoritmo B apresenta desvios baixos relativamente ao valor óptimo.

Quando se compara o desvio médio do Algoritmo B para dimensões pequenas e grandes verifica-se que, independente do valor do parâmetro α, a percentagem de desvio é superior para grandes dimensões. Esta situação é mais evidente quando α=0 (ver Gráfico 52).

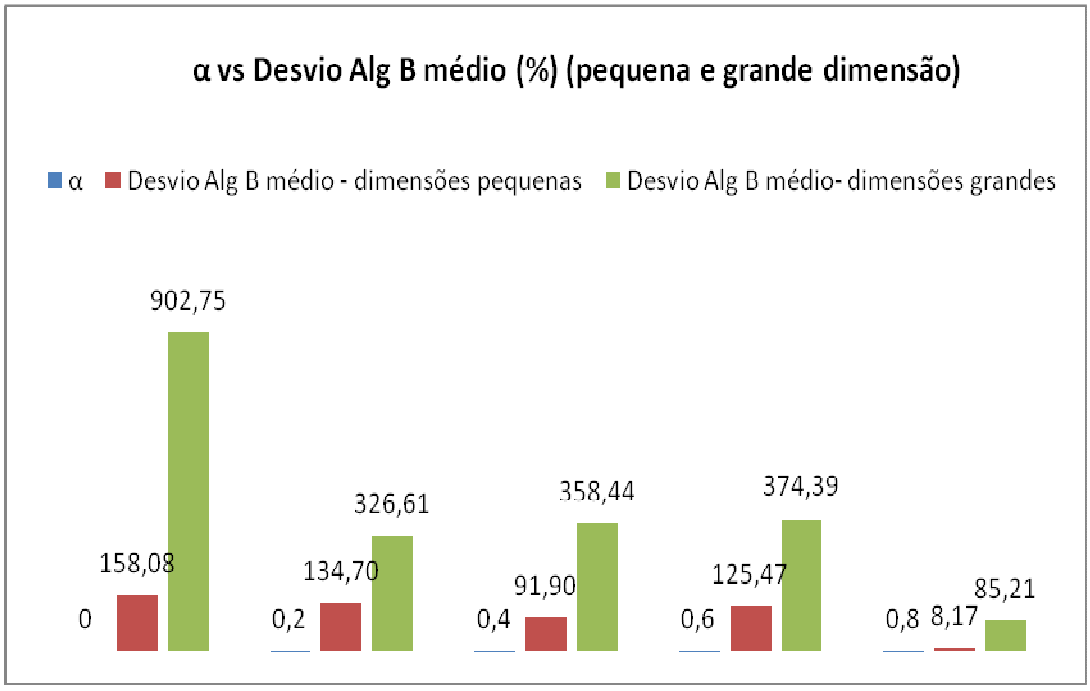


Gráfico 52

8 Conclusões

Efectuou-se uma abordagem do problema do caminho mais curto com uma restrição de capacidade. Foi proposto um novo algoritmo, Algoritmo B, baseado na relaxação da restrição de capacidade do problema. Partindo de um caminho inicial, este algoritmo determina sucessivamente vários caminhos até obter uma solução admissível.

Procedeu-se a um estudo computacional, onde foi possível identificar algumas das propriedades das instâncias mais difíceis do PCMCRC, nomeadamente quando:

- a dimensão do problema é grande;
- o grafo é estruturado por níveis e quando os pesos e os custos são gerados aleatoriamente num intervalo, tendo em consideração a distância entre os índices dos nós;
- os custos são obtidos de forma inversa relativamente aos pesos;
- a densidade do grafo é elevada.

Foi, ainda, estudado o desempenho do Algoritmo B em função dos parâmetros utilizados na geração das várias instâncias do PCMCRC. Conclui-se que, na globalidade, este algoritmo apresenta tempos de execução baixos. Relativamente à qualidade da solução obtida verificou-se que, apesar de, em geral, o valor da solução apresentar um desvio baixo relativamente ao valor óptimo do problema, em algumas instâncias com características particulares esse desvio foi enorme.

Apesar de não ter sido feito esse estudo, o Algoritmo B pode ser testado para problema do caminho mais curto com múltiplas restrições. Como trabalho futuro seria importante testar o Algoritmo B para problemas com várias restrições de capacidade e para diferentes parâmetros (com alterações, por exemplo, ao nível do número de iterações e variações do parâmetro λ).

Bibliografia

AHUJA, Ravindra K., MAGNANTI, Thomas L. e ORLIN, James B., 1993, *Network Flows*, Prentice Hall

AVELLA, Pasquale, BOCCIA, Maurizio e SFORZA, Antonio, 2004, *Resource Constrained Shortest Path Problems in Path Planning for Fleet Management*, Journal of Mathematical Modelling and Algorithms, 3, pp. 1-17

BOLAND, Natashia, DETHRIDGE, John e DUMITRESCU, Irina, 2005, *Accelerated label setting algorithms for the elementary resource constrained shortest path problem*, Operations Research Letters, 34, pp. 58-68

BURIOL, L.S., RESENDE, M. G. C. e THORUP, M., 2003, *Speeding up dynamic shortest path algorithms*, INFORMS Journal on Computing, 20, pp.191-204

DAHL, Geir e REALFSEN, Bjornar, 2000, *Curve approximation and constrained shortest path problems*, Networks, 36, pp. 1-8

DESROCHERS, M, DESROSIERS, J e SOLOMON, M, 1992, *A new optimization algorithm for the vehicle routing problem with time windows*, Operations Research, 40, pp. 342-354

ELIMAM, A. A. e KOHLER, David, 1997, *Two engineering applications of a constrained shortest-path model*, European Journal of Operational Research, 103, pp. 426-438

GAREY, M. e JOHNSON, D., 1979, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, New York.

GUERRIERO, F. e MUSMANNO, R., 2001, *Label Correcting Methods to Solve Multicriteria Shortest Path Problems*, Journal of Optimization Theory and Applications, 111, nº3, pp. 589 – 613

HANDLER, Gabriel Y. e ZANG, Israel, 1980, *A Dual Algorithm for the Constrained Shortest Path Problem*, Networks, 10, pp. 293-310

LAPORTE, Gilbert, NOBERT, Yves e DESROCHERS, Martin, 1985, *Optimal Routing under Capacity and Distance Restrictions*, PhD Thesis

MEHLHORN, K. e ZIEGLEMANN, M., 2001. CNOP – A Package for Constrained Network Optimization, LNCS 2153, pp. 17–31

MINOUX, M. e RIBEIRO, C., 1985, *A heuristic approach to hard constrained shortest path problems*, Discrete Applied Mathematics, 10, pp. 125-137

MINOUX, M. e RIBEIRO, C., 1985, *Solving hard constrained shortest problems by Lagrangean relaxation and branch-and-bound algorithms*, Methods of Operations Research, 53, 304-316

PISINGER, David, 2005, *Where are the hard knapsack problems?,* Computers & Operations Research, 32, pp. 2271-2282

RIGINI, Giovanni e SALANI, Matteo, 2008, *New dynamic programming algorithms for the resource-constrained elementary shortest path problem,* Networks, 51, pp. 155-170

RIGINI, Giovanni e SALANI, Matteo, 2006, *Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints*, Discrete Optimization, 3, pp. 255-273

SOUSA, Amaro Fernandes de, 2006, *Modelos de Problemas de Optimização em Redes de Telecomunicações*, Universidade de Aveiro

WOLSEY, Laurence A., 1998, *Integer Programming*, Jonh Wiley & Sons

XUE, Guoliang, 2000, *Primal-Dual Algorithms for Computing Weight-Constrained Shortest Paths and Weight-Constrained Minimum Spanning Trees,* IEEE, pp. 271-277

ZIEGLEMANN, Mark, 2001. *Constrained Shortest Paths and Related Problems*, PhD Thesis

ZHU, Xiaoyan, 2005, The Dynamic, *Resource-constrained shortest path problem on an acyclic graph with application in column generation and a literature review on sequence-dependent scheduling*, PhD Thesis